0

April 1989

**TEST CASE STUDY:**

# Estimating the Cost of Ada Software Development

*Prepared for:*

U.S. Air Force Cost Center (AFCSTC)
Arlington, Virginia 22202

U.S. Army Cost and
Economic Analysis Center (USACEAC)
Washington, D.C. 20324-0200

Ada Joint Program Office (AJPO)
Arlington, Virginia 22202

*Prepared by:*

IIT Research Institute
4600 Forbes Boulevard
Lanham, Maryland 20706-4324

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| | | |
|---|---|---|
| **REPORT DOCUMENTATION PAGE** | | READ INSTRUCTIONS BEFORE COMPLETEING FORM |
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) TEST CASE STUDY: ESTIMATING the Cost of Ada Software Development | | 5. TYPE OF REPORT & PERIOD COVERED |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) | | 8. CONTRACT OR GRANT NUMBER(s) F 19628-89-C-061 |
| 9. PERFORMING ORGANIZATION AND ADDRESS IIT RESEARCH INST. | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Ada Joint Program Office United States Department of Defense Washington, DC 20301-3081 | | 12. REPORT DATE APRIL, 1989 |
| | | 13. NUMBER OF PAGES |
| 14. MONITORING AGENCY NAME & ADDRESS (If different from Controlling Office) AJPO | | 15. SECURITY CLASS (of this report) UNCLASSIFIED |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20 If different from Report)

UNCLASSIFIED

18. SUPPLEMENTARY NOTES

19. KEYWORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1473 EDITION OF 1 NOV 65 IS OBSOLETE
1 JAN 73 S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# TEST CASE STUDY:

# ESTIMATING THE COST OF ADA SOFTWARE DEVELOPMENT

April 1989

Prepared For:

U.S. Air Force Cost Center (AFCSTC)
Arlington, Virginia 22202

U.S. Army Cost and Economic Analysis Center (USACEAC)
Washington, D.C. 20324-0200

Ada Joint Program Office (AJPO)
Arlington, Virginia 22202

Prepared By:

IIT Research Institute
4600 Forbes Boulevard
Lanham, Maryland 20706-4324

## ACKNOWLEDGEMENT

This report is published in the interest of scientific and technical information exchange. It represents a small window in time with respect to the state-of-the-art in software cost estimating. As Ada development methodologies continue to evolve, model developers are refining their products regularly. Readers are advised to consult model vendors to discuss capabilities and new model releases. IIT Research Institute (IITRI) gratefully acknowledges the support of the model developers in assembling the information for this report. We appreciate the considerable cooperation of software developers who provided project data and the contributions of the following individuals: Neal Brenner, Joseph Dean, Scott Belford, and Bob Arace.

## TABLE OF CONTENTS

## TABLE OF CONTENTS (Continued)

## LIST OF APPENDICES

## LIST OF TABLES

TABLE OF CONTENTS (Continued)

## TABLE OF CONTENTS (Continued)

## LIST OF FIGURES

## GLOSSARY OF TERMS

| | |
|---|---|
| AFB | Air Force Base |
| AFCSTC | Air Force Cost Center |
| AJPO | Ada Joint Program Office |
| APM | Ada Process Model |
| APSE | Ada Programming Support Environment |
| ASLOC | Ada Source Lines of Code |
| BMO | Ballistic Missile Office |
| C/A | Contract Award |
| CDR | Critical Design Review |
| CUT | Code and Unit Test |
| CEI | Computer Economics, Inc. |
| CM | Configuration Management |
| ESD | Electronic Systems Division |
| FQR | Formal Qualification Review |
| FQT | Formal Qualification Test |
| HOL | High Order Language |
| IITRI | IIT Research Institute |
| IOC | Initial Operations Capability |
| MIS | Management Information System |
| MODP | Modern Development Practices |
| OOD | Object Oriented Design |
| O&S | Operation and Support |
| OT&E | Operational Testing and Evaluation |
| PCA | Physical Configuration Audit |

| | |
|---|---|
| PDR | Preliminary Design Review |
| PQT | Preliminary Qualification Test |
| QA | Quality Assurance |
| RCI | Reifer Consultants, Inc. |
| SDR | System Design Review |
| SLOC | Source Lines of Code |
| SPR | Software Productivity Research, Inc. |
| SRR | System Requirements Review |
| SSR | Software Specification Review |
| TRR | Test Readiness Review |
| USACEAC | United States Army Ccst and Economic Analysis Center |
| USAF/ESD | United States Air Force/Electronic Systems Division |

## EXECUTIVE SUMMARY
## TEST CASE STUDY: ESTIMATING THE COST OF ADA SOFTWARE DEVELOPMENT

The Ada Joint Program Office (AJPO) has had a number of inquiries on how to estimate the cost of Ada projects, and specifically whether current software cost models that are non-Ada specific can be successfully used to estimate the cost of Ada development efforts. No independent data currently exists that addresses the fidelity of cost models to predict Ada software costs accurately.

IIT Research Institute (IITRI) has performed a study for the U.S. Air Force Cost Center (AFCSTC) and the U.S. Army Cost and Economic Analysis Center (USACEAC), under sponsorship of the AJPO, to assess the accuracy of current cost models for Ada software cost estimation. The study focussed on six cost estimation models and their philosophies regarding Ada. The bases for these models present different views of the model developers relative to the costing issues.

The guiding principle for model selection for inclusion in the study was the availability of models to the AFCSTC, USACEAC, and IITRI. Models reviewed were as follows:

| <u>Ada-Specific Models</u> | <u>Non-Ada-Specific Models</u> |
|---|---|
| 1. Ada COCOMO (IOC) | 1. PRICE S (188230) |
| 2. SoftCost-Ada (1.3) | 2. SYSTEM-3 (1.03) |
| | 3. SPQR/20 (1.2) |
| | 4. SASET (1.5) |

The version of each model reviewed is indicated in parenthesis beside the model name.

An essential part of the research is a test case study in which the cost models were applied to a database of eight completed Ada projects. Project questionnaires were completed by the developers. This information was used to derive inputs for each model. Emphasis was placed on providing a consistent set of inputs across all models. In addition, models were applied using nominal (average) values for input ratings while providing actual project values for size, application type, programming language, and other model inputs that must be estimated early in the life cycle and for which there is no associated average value. The nominal inputs reflect the level of knowledge about a new project prior to contract award. Resultant analyses were based on a comparison of each model's schedule and effort projection and nominal run results to the actual project resources expended by the software developer.

Results were evaluated for accuracy and consistency for each of the following six categories:

1. Overall effort
2. Overall schedule
3. Government contracts
4. Commercial contracts
5. Command and control applications
6. Tools and environment applications.

Tables 1 and 2 respectively summarize the test case study results for derived and nominal inputs. For each evaluation criteria, the two models that had the highest performance ratings are listed. Model results were also evaluated based on the project's design approach and personnel experience with Ada. Model performances could not be correlated to the language considerations of the models.

The test case study results demonstrate the benefits of cost models that assist the estimator in predicting resource requirements for a new development. The results do not validate the need for Ada-specific models. Although SoftCost-Ada was most accurate overall, non-Ada-specific models were comparable in terms of accuracy and consistency.

The results do recommend that users consider the following to determine which models should be applied to estimate Ada software costs:

- **Assess how much information is available about the project and the developing organization.** Application of models to both regular and nominal runs in the study showed that model performances varied with differing amount of project information. Some performed better with minimum information while others performed better with detailed information.

- **Consider the customer.** Model performances were evaluated based on the type of contract. Some models were more effective when applied to government contracts while others were more accurate for estimating commercial contracts.

- **Consider the type of application.** Projects targeted in the test case study consisted of three different types of applications: command and control (4 projects), tools/environment (3 projects), and avionics (1 project). An analysis of results based on application type revealed that models that were most accurate on command and control applications were not as accurate for tools and environment applications.

TABLE 1

SUMMARY TEST CASE STUDY RESULTS:
BEST TWO PERFORMANCES IN EACH CATEGORY

| Evaluation Criteria | Model | Performance (Within 30%) | Range |
|---|---|---|---|
| Overall Accuracy of Effort | SoftCost-Ada<br>SASET | 4 out of 7<br>4 out of 8 | 0% to 13%<br>-29% to -29% |
| Overall Accuracy of Schedule | SYSTEM-3<br>PRICE S | 4 out of 8<br>3 out of 8 | -27% to - 7%<br>3% to 18% |
| Overall Consistency of Effort | SYSTEM-3<br>PRICE S | 5 out of 8<br>5 out of 8 | -14% to 28%<br>-26% to 22% |
| Overall Consistency of Schedule | SYSTEM-3<br>PRICE S | 5 out of 8<br>5 out of 8 | 0% to 28%<br>-29% to 28% |
| Model Accuracy on Government Contracts | SASET<br>COSTMODL | 3 out of 4<br>2 out of 4 | - 7% to 29%<br>-25% to - 1% |
| Model Consistency on Government Contracts | SYSTEM-3<br>PRICE S | 4 out of 4<br>3 out of 4 | -14% to 28%<br>-26% to 0% |
| Model Accuracy on Commercial Contracts | SoftCost-Ada<br>SPQR/20 | 3 out of 3<br>1 out of 4 | 0% to 6%<br>-22% |
| Model Consistency on Commercial Contracts | SoftCost-Ada<br>PRICE-S | 3 out of 3<br>2 out of 4 | -13% to - 8%<br>- 1% to 22% |
| Model Accuracy on Command & Control Applications | SASET<br>SPQR/20 | 3 out of 4<br>3 out of 4 | - 7% to 29%<br>-22% to 19% |
| Model Consistency on Command & Control Applications | PRICE S<br>SASET | 4 out of 4<br>3 out of 4 | -26% to 0%<br>-15% to 1% |
| Model Accuracy on Tools/Environment Applications | SoftCost-Ada<br>SASET | 2 out of 2<br>1 out of 3 | 0% to 2%<br>-29% |
| Model Consistency on Tools/Environment Applications | SoftCost-Ada<br>SYSTEM-3 | 2 out of 2<br>1 out of 3 | -13% to -11%<br>28% |

TABLE 2

SUMMARY TEST CASE STUDY RESULTS FOR NOMINAL RUNS:
BEST TWO PERFORMANCES IN EACH CATEGORY

| Evaluation Criteria | Model | Performance (Within 30%) | Range |
|---|---|---|---|
| Overall Accuracy of Effort | SASET<br>SYSTEM-3 | 4 out of 8<br>3 out of 8 | -24% to 29%<br>-17% to 28% |
| Overall Accuracy of Schedule | SPQR/20<br>PRICE S | 6 out of 8<br>4 out of 8 | -23% to 28%<br>-26% to 21% |
| Overall Consistency of Effort | COSTMODL<br>SoftCost-Ada | 3 out of 6<br>3 out of 7 | -23% to 30%<br>0% to 28% |
| Overall Consistency of Schedule | SPQR/20<br>PRICE S | 6 out of 8<br>5 out of 8 | -28% to 20%<br>-28% to 29% |
| Model Accuracy on Government Contracts | SASET<br>PRICE S | 3 out of 4<br>2 out of 4 | - 7% to 29%<br>-14% to - 8% |
| Model Consistency on Government Contracts | SoftCost-Ada<br>SYSTEM-3 | 3 out of 4<br>3 out of 4 | 0% to 28%<br>-26% to 13% |
| Model Accuracy on Commercial Contracts | COSTMODL<br>SoftCost-Ada | 1 out of 2<br>1 out of 3 | -24%<br>14% |
| Model Consistency on Commercial Contracts | SASET<br>SPQR/20 | 1 out of 4<br>1 out of 4 | 7%<br>-14% |
| Model Accuracy on Command & Control Applications | SASET<br>SYSTEM-3 | 3 out of 4<br>3 out of 4 | - 7% to 29%<br>-17% to 28% |
| Model Consistency on Command & Control Applications | SASET<br>SoftCost-Ada | 3 out of 4<br>2 out of 4 | -24% to 7%<br>12% to 28% |
| Model Accuracy on Tools/Environment Applications | SASET | 1 out of 3 | -24% |
| Model Consistency on Tools/Environment Applications | | 0 | |

# 1.0 INTRODUCTION

## 1.1 BACKGROUND

The transition to a new technology like Ada takes considerable time, commitment, and resources. It is generally held that during the short-term, Ada software projects will cost more as a result of inexperienced staff, immature software support tools, and design and packaging costs for developing reusable software. These additional costs will be offset in the long-term as developers begin to derive the benefits of reuse and personnel spend less effort maintaining code that is less error prone.

Perceptions vary widely on how to estimate the cost of Ada projects. Since Ada was established as the single common computer programming language for all new developments and major upgrades of Defense systems, two Ada-specific cost estimating models have been developed: SoftCost-Ada and Ada COCOMO. In addition, cost drivers in several other models have been adjusted so that they could support near-term and long-term Ada cost estimating needs.

The bases for these models present the different views of the model developers relative to costing issues. An Ada study conducted by Reifer Consultants, Inc. (RCI), developers of SoftCost-Ada, during 1986 [REIF87] concluded that existing software cost models, largely based on non-Ada development efforts, do not accurately account for Ada's risks and actual experiences. Power laws for Ada are different from other high order languages (HOLs) and their effects cannot be masked by adjusting cost estimating relationships. Calibration is different because of the need to account for the Ada learning curve and software reuse.

Dr. Barry Boehm of TRW, Inc., a co-developer of Ada COCOMO, agrees that Ada and the process for developing Ada software impacts the manner in which development cost is estimated. While the existing COCOMO has done reasonably well in estimating Ada software costs, experience to date at TRW has indicated that a version of COCOMO specifically tailored to Ada would probably do better. The effects of using Ada and practices relating to the Ada Process Model lead to new cost estimating relationships that accommodate the resulting cost and schedule effects.

Other independent studies conducted by developers of SPQR/20, SYSTEM-3, and PRICE S recommend no significant modifications. These studies indicate that present models are adequate for Ada estimation with the provision that some input parameters are slightly modified based on Ada experience. User's of these models select their inputs with regard to a calibrated baseline that represents their typical development. The input responses reflect the ways in which the Ada project differs from a reference baseline. Learning curves and reuse factors are not treated as Ada-unique phenomena.

The Ada Joint Program Office (AJPO) has had a number of inquiries on how to estimate the cost of an Ada project and whether current software cost models that are non-Ada specific can be successfully used to estimate the cost of Ada development efforts. Currently, there is no independent data that validates the need for Ada-specific models or justifies the use of present models that are non-Ada-specific. Part of the problem in evaluating the accuracy of a model (comparing the model's projections to the actual cost of a development) is that many contracts are awarded at a firm fixed price. Thus, unless a concerted effort is made to track the real cost and scope of the effort, no assessment can be made with regard to the fidelity of the model to predict Ada software costs accurately.

## 1.2 OBJECTIVE

The objectives of this report are centered around the major issues regarding Ada software development cost estimation. To assist the cost analyst in arriving at reasonable Ada cost estimates, this report will focus on the following objectives:

(1) Determine if software cost models that are non-Ada-specific can be successfully used to estimate the cost of Ada development.

(2) Provide a comprehensive review of selected cost models with regard to Ada-specific cost drivers and issues discussed in literature. Evaluate model performance in light of the language considerations of the models to determine any correlation.

(3) Identify which models provide reasonable results when used to estimate Ada development cost.

An essential part of the research is a test case study in which selected cost models were applied to a database of eight completed Ada projects. This information was used to derive inputs for each model. Emphasis was placed on providing a consistent set of information for each model. In addition, models were applied using nominal (average) values for input ratings while providing actual project values for model inputs which have no associated average value. Model performance was evaluated based on a comparison of each model's projected schedule and effort to the actual resources expended by the software developer for each project. The results of the test case study are the bases for recommending a preferred approach to estimating Ada software costs. A detailed description of the test case study, results, and conclusions is included in this report.

## 1.3 REPORT OUTLINE

The organization of this report corresponds to the defined tasks outlined in the study objective:

- Description of the Models - Section 2

  This section provides an overview of the philosophy behind each model with regard to Ada versus other high order languages. Each model is described with regard to several potential cost drivers and issues that are believed to have a different impact on Ada versus non-Ada projects. A matrix that demonstrates the language considerations of current models is included in this section.

- <u>Application of Selected Cost Models to Ada Projects</u> - Section 3

  This section provides an overview of the Ada projects targeted in the test case study. The methodology used to collect and validate project data, derive the model input, and subsequently interpret the results of the models applications is presented.

- <u>Analysis of Ada Project Data</u> - Section 4

  This section describes the sensitivity analysis performed to validate cost drivers previously identified and uncover new cost drivers within an application domain.

- <u>Conclusion</u> - Section 5

  This section summarizes findings from each of the defined tasks described in the previous sections. An approach to estimating Ada software development costs is recommended that considers the differences in the model philosophies, cost drivers, and performance in view of the domain of Ada projects targeted in the test case study. Lessons learned on data aquisition and data analysis are also included in this section.

## 1.4 <u>TERMINOLOGY</u>

The following definitions pertain to the terminology used within this study.

<u>Ada:</u> A programming language that was designed and developed by the United States Department of Defense (DoD) to establish one common programming language for all its applications and incompatible dialects used by its programmers.

<u>Consistency</u>: The comparison of the project's actual effort and schedule duration to the model's estimated effort and schedule after a computed mean value has been applied to each model estimate.

<u>Incremental Development:</u> A software development paradigm characterized by implementing and testing one part of a system, then implementing and testing another, one by one, until the system is complete.

<u>Instantiation</u>: The process of representing an abstraction by a concrete example. In Ada, the instantiation of a generic creates a new subprogram or package that can be used.

<u>Model:</u> The underlying mathematical equation or integrated set of equations. For example, COCOMO is a single equation cost model; PRICE S is an integrated system of equations and operations.

Normalization: The process of conforming or reducing to a standard answer.

Object Oriented Design: A partitioning of a software system according to classes or objects, not functions.

Package: An implementation of a given model. For example, SECOMO and COSTMODL are both implementations of the COCOMO model.

Pilot Development: An initial program limited in either scope or functionality of the final system to test the feasibility of implementing a new capability.

Program Library: An organized repository of reusable code.

Prototyping: The process of developing an early model that represents the actual product.

Relative Error: The comparison of estimated effort or schedule duration to the actual effort or schedule using the following measure:

relative error = (estimated effort - actual effort)/actual effort

Spiral Development: A software development paradigm characterized by successively refined understandings of the problem and successively refined prototypes of a software solution to the problem.

Structured Analysis: The activity of deriving a functional model of the requirements for a system.

Waterfall Development: A software development paradigm characterized by a progression through a series of steps (Requirements, Design, Coding, Testing), each followed by some form of verification (Software Requirements Review, Critical Design Review, etc.).

This page is intentionally left blank.

## 2.0 DESCRIPTION OF THE MODELS

### 2.1 ADA OVERVIEW

Models that are included in this study were selected based upon their availability to either the AFCSTC, USACEAC, or IITRI. Model vendors were not solicited for their participation. In addition, applicability of the models to the projects targeted in the test case study was a further consideration. For example, Estimacs is a model used for estimating the cost of business-type applications. Projects targeted in this study are primarily command and control and tools/environment applications. Although Estimacs was available to the AFCSTC, it was not included in this study because of its applicability.

Software cost estimation models were reviewed with regard to potential cost drivers and issues that have a different impact on Ada versus non-Ada projects. The review focused on two Ada-specific models used to estimate development costs for projects in which Ada is the primary software language, and four non-Ada-specific models appropriate for application to Ada projects and other HOLs:

| Ada-Specific Models | Non-Ada-Specific Models |
|---|---|
| 1. Ada COCOMO (IOC) | 1. PRICE S (188230) |
| 2. SoftCost-Ada (1.3) | 2. SYSTEM-3 (1.03) |
| | 3. SPQR/20 (1.2) |
| | 4. SASET (1.5) |

The package version of each model reviewed is indicated in parenthesis beside the model name. The COSTMODL package, Version 5.0, which implements the complete Initial Operational Capability (IOC) version of the Ada COCOMO model, was used for the purposes of this study. COSTMODL (5.0) implements the Ada COCOMO model introduced by Dr. Boehm at the 1987 COCOMO User's Group Conference. It does not include the enhancements to the Ada model that Dr. Boehm introduced at the 1988 COCOMO Users' Group Conference. (The 1988 enhancements are discussed in Appendix F).

The purpose of the review was to produce a summary of the Ada technology considerations for each model. The results of the review are presented in the following sections. Section 2.1 will focus on each model while section 2.2 focuses on each potential cost driver and issue.

Model overviews presented in this section include the following information (when available):

- Philosophy with regard to Ada
- Recommended responses to certain model inputs for Ada
- Ada projects used to develop and validate the model
- Clientele who use the model for Ada projects.

General information regarding hardware, availability, and scope of coverage is also provided for each model in the report appendices.

### 2.1.1 Ada COCOMO

The Ada COCOMO model is a tailored version of the standard COCOMO model developed by Dr. Barry W. Boehm of TRW's Software Information Systems Division and described in full detail in Boehm's book, Software Engineering Economics [BOEH81].

Although the existing COCOMO model has done reasonably well in supporting Ada cost estimates, experience to date at TRW has indicated that a version of COCOMO specifically tailored to Ada would probably do better. One of the reasons for developing a new model was to incorporate the TRW Ada Process Model which assumes a phased incremental development process instead of a waterfall development paradigm. Phased incremental development can also be adapted to other programming languages [BOEH87].

Barry Boehm and Walker Royce led development of the Ada version of COCOMO in a process that began in 1985-86 with an initial version of the Ada Process Model and an initial set of hypotheses which dealt with the effect of Ada on standard COCOMO cost drivers. In early 1987, a two-phase Delphi process, involving 10 TRW personnel experienced in Ada, software cost estimation, and large software projects, was employed to refine the functional form of Initial Operations Capability (IOC) Ada COCOMO, and to determine an initial set of revised cost driver multipliers and exponents [BOEH87].

In mid-1987, this initial model was calibrated, using two completed TRW Ada projects which had been developed using full DoD software acquisition standards. Subsequently, the model development process and product were subjected to an independent TRW review, resulting in some further refinements. The resulting IOC Ada COCOMO Model was presented at the Third COCOMO User's Group Meeting, at the Software Engineering Institute, in November 1987 [BOEH87].

Differences between the standard COCOMO and Ada COCOMO reflect a philosophy that Ada experiences are like that of any other new language if the development process is implemented using the techniques and milestones of most previous software development projects. Nominal recalibration of required reliability, complexity, and language experience is the impact of the Ada language. The more significant impact is the development process -- which is reoriented to capitalize on Ada strengths and to reinforce more efficient software production methods. These effects include modified exponential scaling equations for estimating nominal development effort, schedule, and maintenance effort. Phase distributions for effort and schedule are revised.

The overall functional form, most of the effort multipliers, adaptation equations, activity distribution tables, and the use of

Annual Change Traffic for maintenance estimation remain the same in both standard and Ada COCOMO. The IOC Ada COCOMO model includes some general improvements which also apply to standard COCOMO [BOEH87]:

- There is a wider range of ratings and effects due to software tools (TOOL) and turnaround time (TURN).
- Virtual machine volatility (VIRT) are split into host machine (VMVH) and target machine (VMVT) effects.
- The added cost due to schedule stretchout is eliminated.
- Cost drivers are added to cover the effect of classified projects (SECU) and development for software reusability (RUSE).
- A capability is added to estimate the costs and schedules involved in using a phased incremental development process.

Other changes relate to the effects that are specific to Ada and the TRW Ada Process Model [BOEH87]:

- Cost and schedule effects specific to Ada comprise the following changes to standard COCOMO:

    1. Multiplier penalties for higher levels of required reliability (RELY) and complexity (CPLX) are reduced.
    2. There is a wider range of ratings and effects due to programming language experience (LEXP).
    3. There are new Ada-oriented instructions for counting lines of code and reusing software in Ada.

- Effects of using the TRW Ada Process Model, which reorients portions of the software development process, result in the following cost and schedule effects:

    1. Exponential scaling equations for nominal development effort, development schedule, and nominal maintenance effort are revised.

2. The range of effects of modern programming practices (MODP), analyst capability (ACAP), and programmer capability (PCAP) have been extended.

3. Phase distributions of effort and schedule are revised.

The general improvements to COCOMO and the language-specific modifications apply to the standard waterfall model. They also apply to a single increment use of the Ada Process Model along with the changes due to the Ada Process Model. For a phased incremental development, changes due to the Ada Process Model apply [BOEH88].

Although Ada COCOMO is not available in automated form directly from Boehm or TRW, there are several automated implementations commercially available from other organizations. These include NASA Johnson Space Center's COSTMODL; COSTAR developed by Softstar Systems; GECOMO developed by GEC Software; and an IOC Ada COCOMO version implemented by the Ballistic Missile Office (BMO-ACS). Appendix A lists specific points of contact for Ada COCOMO implementations. When inquiring about a package implementation of Ada COCOMO, it is important to note which version of the model the package implements, (i.e., 1987 IOC version, 1988, etc.) and whether the package implements the incremental development model.

A detailed description on how to generate an estimate with the Ada COCOMO model is provided in Appendix F. Note that **current estimating equations only apply to Intermediate COCOMO, embedded-mode software projects** (see pages 78 - 83, Software Engineering Economics [BOEH81]). Equations have not been developed for organic or semidetached modes of Intermediate COCOMO. Basic COCOMO effort and schedule equations are also not included in the IOC version of Ada COCOMO.

Because Ada COCOMO is an initial operating capability calibrated to only two completed Ada projects, use of the model is not widespread. TRW uses the model in parallel with standard COCOMO and other Ada cost models to increase perspective on Ada cost estimates. TRW will continue Ada project data collection for further refinement and calibration of Ada COCOMO [BOEH87].

2.1.2 <u>SoftCost-Ada</u>

SoftCost-Ada is a derivative of RCI's SoftCost-R model, which generates schedule and cost estimates for systems implemented in conventional programming languages. When SoftCost-R was initially run on Ada project data, the results were inconsistent. Attempts were made to modify the model for Ada projects, but the language marked such a shift in the philosophy of programming languages that new equations were developed specifically for it. Four cost drivers were identified for costing an Ada project that were not addressed in SoftCost-R [RCI88]:

- Ada Usage factor: the percentage of code expected to be written in Ada.
- Degree of Real-Time: the relative complexity of the tasking to be performed in the system.
- Degree of Reuse: the percentage of new code expected to be packaged for reuse, both internal and external to the project.
- Resource Availability: the availability of resources, such as staff, tools, equipment, etc., within the developer's organization.

Cost drivers addressed in SoftCost-R which have a significant impact on Ada developments include [RCI88]:

- Degree of Standardization
- Use of Modern Software Methods
- Use of Software Tools/Environments
- Software Tools/Environments Stability.

Data used to develop the SoftCost-Ada model consisted of approximately 30 software projects developed by 12 different organizations within five aerospace firms during the period spanning 1982 through 1987. Projects by application area included automation, command and control, commercial avionics, embedded, environment/tools, telecommunications, and simulator software. Today, RCI's database consists of more than 90 projects, which represent more than 25 million lines of Ada source code [RCI88]. Clientele who use the model for Ada projects include Rockwell, Shell Oil, Singer Link, Westinghouse, AAI, Contel, Nippon Telegraph and Telephone (Japan), U.S. Air Force, DND (Canada), Philips (Sweden), and Ferranti (England).

### 2.1.3 PRICE S

Note: The following description is condensed from "Ada Estimating- - A PRICE S Profile" prepared by Dr. Robert E. Park, PRICE Systems [Park89]

PRICE S approaches estimating for Ada in the same way that it approaches estimating for other development environments. Costs and schedules are driven more strongly by product characteristics and developer practices than by development languages. However, PRICE S does recognize that different source languages have different support environments and different productivity characteristics. Some languages, like Ada, even imply different management methodologies.

When Ada is specified, PRICE S does three things:

- It uses relationships appropriate to Ada to translate the number of source lines of code into the model's internal measure for product size.
- It uses an Ada-specific technology submodel to estimate the trends in productivity improvement that occur as a function of time.
- It distributes the activities of designers, programmers, systems engineers, and program managers more toward the early phases than it does for other development environments.

Otherwise, all parameters in PRICE S are used and evaluated just as in other software estimates. Estimators are not asked to change tools or procedures in order to estimate Ada projects.

When configuring PRICE S for Ada estimating, users pay special attention to parameters that take on values different from those used to describe their reference baselines. For example, if the reference projects contain no Ada software, then the difference expected with Ada must be spelled out. PRICE S recommends that the estimator examine how the project being estimated may differ from the reference baseline with regard to the following factors:

- Resource attributes:
  - Productivity factors
  - Schedule distribution
  - Cost element distribution
  - Labor rates (if different personnel are used and monetary costs are desired)
- Personnel attributes:
  - Team experience
  - Team skill
- Product familiarity:
  - Extent of experience with similar software
  - Extent of experience with software of similar size
- Software tools
- New programming language
- New computer hardware
- Unusual levels of virtual machine experience
- Unusual levels of virtual machine volatility
- Unusual requirements to produce reusable code (generics, for example)
- Introduction of new practices for software design and programming

These are the most common ways in which Ada projects can differ from their predecessors. PRICE S parameters permit the responses for each assessment to be incorporated into an estimate.

Estimating with PRICE S is easiest when completed Ada projects are available for calibration. Here, organizational productivities, scheduling practices, and cost distributions can be measured. These measurements can be used to configure the model to fit demonstrated Ada performance. Differences between new projects and calibrated references then become smaller and easier to judge than when comparisons must be made against non-Ada baselines.

In all cases, PRICE S philosophy recognizes that software developers differ and that development environments are dynamic. The philosophy stresses that the most reliable basis for an estimate is that of extrapolation from demonstrated performance. As tools and management methods evolve to take advantage of Ada, users of PRICE S will find some parameters taking on different values, but the procedures for estimating will not need to change.

### 2.1.4 SASET

In 1986 while under contract to the Naval Center for Cost Analysis, Martin Marietta conducted a study of existing software estimating and sizing models. In its findings, Martin Marietta determined that there was a need for a forward-chaining, rule-based expert system utilizing a hierarchically structured knowledge database to provide functional software sizing values, optimal development schedule, and associated manloading outputs. Martin Marietta developed the computerized model Software Architecture Sizing, and Estimating Tool (SASET) as a prototype in 1987 that would satisfy these requirements [CHEA89].

SASET is a rule-based expert system that utilizes a "three-tiered" approach for system identification. Tier 1 - System Environment-requests information regarding the class of software, programming language, development schedule, development locations, and other factors which describe the development environment. Output from this tier are factors used in subsequent processing to derive budget and schedule estimates [MART88].

Tier 2 - Software Functionality - automates the process of sizing by analogy. The user indicates, through a process of functional decomposition, those software functions to be contained in the developed software system. A nominal SLOC estimate obtained from the SASET database for each of the individual functions is adjusted based upon the responses provided for each software element. The information:

- o Perceived complexity,
- o Degree of new development,
- o Development language (HOL or Assembly), and
- o Hosting CPU

is used to adjust the function's base sizing estimate and derive a lines of code value. The tier 2 sizing estimate is in equivalent FORTRAN lines of code excluding comments. The user may also enter sizing informatin directly rather than identify specific software functions [MART88].

Tier 3 - System Complexity - requires the user to rate 20 complexity issues on a scale of 1 (very complex) to 4 (simple). A weighting factor is assigned to each response. The product of the weighting factors represents the impact to the software development budget and schedule [MART88].

A fourth tier is used to estimate manloading over the entire maintenance life cycle.

Martin Marietta software development data consisting of more than 300 completed projects and some selected Navy data were used to develop the SASET model [CHEA89]. SASET is meant to be used to estimate development in Assembly, Ada, or any HOL. However, the model does not differentiate between Ada and non-Ada development in its input responses except in the following instances [MART88]:

- o The tier 3 input for complexity of the programming language has a direct effect on the productivity of the scftware programmers. The selections that SASET provides penalize Ada projects for their "complex rigidly structured constructs (training usually required)."

- o The tier 1 input for primary software language has an effect on the software development effort-schedule due to the robustness of the software language and the type/capability of the system being developed. The tier 1 model input allocates software languages into the following four classes:

    1. Ada
    2. LISP, PROLOG, C, Assembly
    3. Traditional HOLs: FORTRAN, Pascal, JOVIAL
    4. User languages, Simple Non-structured HOLs and Test Sequence languages.

SASET was initially developed as a manual model to MIL-STD-483, 490, 1679, and 1521A requirements. It has been recommended that the computerized model be updated to also reflect a DoD-STD-2167A development. In doing so, the following Ada items would be addressed [CHEA89]:

- o Modern Software Engineering Practices
- o Reusability
- o Ada Language Features
- o APSE Tools (Ada Programming Support Environment).

Clientele who use the SASET model for Ada projects are the Naval Center for Costs Analysis and the Air Force Cost Center.

## 2.1.5 SPQR/20

Software Productivity, Quality, & Reliability/20 (SPQR/20) is a cost and quality estimation model for planning software development and maintenance activities. The model was developed by T. Capers Jones of Software Productivity Research, Inc. (SPR) based on historical data from more than 3,000 software projects and more than 200 organizations [SPR88]. The database currently includes approximately 50 Ada projects.

SPQR/20 views Ada like other HOLs. Inputs are dependent upon the particular application. SPQR/20 assigns a default value of 4.5 to Ada for the numeric level of the language relative to assembler which the user is able to change.

SPQR/20 differentiates between several different development paradigms including waterfall method, incremental build, and spiral development. The significant paradigm employed by SPQR/20 is that of pattern matching. The mathematical modeling Jones uses for the tool is based on "circa" 3,500 projects including military real-time and embedded systems, and MIS projects. The algorithms that drive the estimates are derived from this database which is constantly being updated [SPR88].

## 2.1.6 SYSTEM-3

SYSTEM-3 is a cost estimation product developed in 1985 by Computer Economics, Inc. (CEI). SYSTEM-3 and its predecessors, JS-1 (1979) and JS-2 (1984), were developed by Dr. Randall W. Jensen, based upon the research of cost estimation by Norden (1970), Putnam (1976), Doty (1977), Jensen (1979), and Boehm (1981) [IITR87]. SYSTEM-3 reflects a philosophy that Ada experiences are like that of any other new language. While the model takes into account language and tool maturity, modern software engineering practices, the learning curve and reusability impacts, these issues are not treated as unique phenomena to the Ada language [CEI88].

The same inputs are required for Ada and non-Ada projects. There are, however, recommended inputs for several parameters that describe Ada development. The recommended minimum (MIN), nominal (NOM), and maximum (MAX) values for Ada, shown in Table 2-1, define where the user is with regard to Ada experience [GAL86].

TABLE 2-1

RECOMMENDED SYSTEM-3 INPUTS FOR ADA DEVELOPMENT

| Input Parameter | MIN | NOM | MAX |
|---|---|---|---|
| 1. Ada Language Complexity | 3 | 3 | 3 |
| 2. Ada Language Experience | 3 | 3 | 4 |
| 3. Ada Virtual Machine Experience | 3 | 3 | 4 |
| 4. Ada Virtual Machine Volatility | 1 | 1 | 2.5 |
| 5. Ada Use of Modern Development Practices | 5.5 | 7.5 | 8 |
| 6. Ada Virtual Machine Complexity | 2 | 2 | 2 |
| 7. Ada Automated Tool Usage | 5 | 6 | 7.5 |
| 8. Ada Turn Around Time * | 0.5 | 0.5 | 0.3 |

* None of the recently completed Ada projects targeted in this study (see section 3.2, Overview of Ada Projects), indicated response times greater than six-minutes (rating was set to 0.1).

Clientele who use the model for Ada projects include the USAF AFSC Divisions (ASD, ESD, BD, AD, BMO), and Department of Transportation (DOT) Canada.

## 2.2 ADA ISSUES

Commercial software cost models were reviewed in light of Ada-specific cost drivers and issues. A primary resource used to identify these issues was a report prepared for the Electronic Systems Division (ESD), Air Force System Command entitled A Plan for Collecting Ada Software Development Cost, Schedule, and Environment Data (CR-0134/1), dated 2 April 1987 [TECO87]. This study identified several potential cost drivers and issues that have different impact on Ada projects versus non-Ada projects. They are:

1. Time spent in design
2. Allocation of costs to phases in the life-cycle
3. Modern software engineering practices
4. Learning curve for the Ada language and tools
5. Reusability requirements
6. Use of certain Ada language features
7. Ada Programming Support Environment (APSE) productivity tools
8. Language and tool maturity
9. Influence of personnel assigned to Ada projects and the fact that the projects are closely monitored (Hawthorne Effect).

The Ada issues were identified from literature and a series of interviews with software modelers, Ada software engineering experts, and Ada project managers [TEC087]. For each model, Ada issues were mapped to corresponding input parameters, if present, and categorized as one of the following:

1. Recalibrated for development process: New cost driver(s) are introduced, exponential scaling equations are revised, or cost driver ratings are revised for a development process which is reoriented for Ada.

2. Recalibrated for Ada: New cost driver(s) are introduced, exponential scaling equations are revised, or cost driver ratings are revised due to features of the Ada programming language regardless of the development process.

3. Cost driver present but not Ada-unique: The issue is reflected by the response of the user to the model's input parameter(s). Cost driver ratings are not recalibrated for Ada. The specified issue is not treated as Ada-unique phenomena.

4. Externally recalibrated by user: Completed Ada projects by the developing organization can be used to configure the model to fit demonstrated Ada performance with regard to the specified Ada issue.

5. No influence. The specified issue has no impact on software development cost.

Table 2-2 provides a summary of the language considerations of each model applied in the test case study. Each Ada issue is further described in the sections that follow.

### 2.2.1 Phase Distribution of Effort

Phase distribution of effort entails the allocation of time throughout the requirements, design, implementation and testing phases of the development cycle. Experts cited in the ESD report said that the effort distribution of an Ada project may differ from projects using other languages because of the emphasis placed on the early stages (requirements and design) of the life-cycle [TECO87]. One factor that will affect the effort distribution is the newness of the language. Because Ada is a relatively new language and contains unique features not found in other languages (generics, tasks, exception handlers), the early phases will take more time in order to correctly learn and use the language properly. Also effecting the phase

TABLE 2-2

LANGUAGE CONSIDERATIONS OF CURRENT SOFTWARE COST MODELS

| | Phase Distribution Of Effort | Modern Software Development | Learning Curve* | Reusability Impact** | Influence of Ada Language Features | Influence of APSE | Language and Tool Maturity (Optimization) | Hawthorne Effort |
|---|---|---|---|---|---|---|---|---|
| ADA COCOMO | RD | RD | RA/RA | NU/NC | RA | NC | NI | NI |
| SOFTCOST ADA | RA | RA | RA/RA | RA/NC | RA | RA | RA | NI |
| PRICE S | ER | ER | NU/NC | NC/NC | NI | NC | NI | NI |
| SASET | ER | NI | NU/NC | NI/NC | NI | NC | NI | NI |
| SPQR/20 | ER | NI | NU/NC | NI/NC | NI | NC | NI | NI |
| SYSTEM-3 | ER | NC | NU/NC | NI/NC | NI | NC | NI | NI |

| | |
|---|---|
| RD | = Recalibrated for a Development Process |
| RA | = Recalibrated for Ada Programming Language |
| NU | = Cost Driver Present but not Ada-Unique |
| ER | = Externally Recalibrated by User |
| NI | = No Influence |
| * | = Learning the Langague/Proficiency in Language |
| ** | = Development of Reusable Components/Use of Reusable Components |

distribution is the newness and complexity of the development process most commonly used on Ada projects, object oriented design. This type of methodology is known to force interdependencies between modules in the requirements and design phases of the development cycle [TECO87]. Ada provides package specifications to set these interdependencies which become the bottom layers of the software. A project can incur a serious penalty if these specifications are changed in later phases due to the effects a change will have on other modules which interact with the changed specification.

Examining the models in light of this issue shows two different views of the effect of Ada on the phase distribution. One view is that the effort distribution should not change because of the Ada language, but should change to fit the observed performance of the developers. This view is found primarily in the non-Ada specific models, namely SASET, SPQR/20, and SYSTEM-3, and PRICE S. These models do not provide an internal effort equation specific to Ada, but they do supply a nominal effort equation that can be recalibrated externally by the user. The other view is that the phase distribution of effort is different for Ada and is reflected in the model's underlying equations. This view is present in the remaining two cost models, Ada COCOMO and SoftCost-Ada.

**Ada COCOMO.** The TRW Ada Process Model reorients portions of the software development process to capitalize on Ada strengths and reinforce more efficient software production methods [BOEH88]. The Process model, which can be adapted to non-Ada projects, is inherent to the Ada version of COCOMO. If the Ada Process model is being implemented, phase distribution of effort will shift from integration and test to design phases. Table 2-3 shows the schedule and effort distribution by phase if there is compliance with the Ada Process Model. A similar breakout of effort for standard COCOMO is shown for comparison. Partial compliance with the Ada Process Model will exhibit a phase distribution in between that shown for standard COCOMO and Ada COCOMO.

TABLE 2-3

COMPARISON OF ADA COCOMO AND STANDARD COCOMO SCHEDULE AND EFFORT DISTRIBUTION BY PHASE.

Standard COCOMO: [Large Embedded]

| Phase Distribution | SSR-PDR | PDR-CDR | CDR-TRR | TRR-FQR |
|---|---|---|---|---|
| Effort | 18 | 25 | 26 | 31 |
| Schedule | 36 | { 36 } | | 28 |

IOC Ada COCOMO: [Large Embedded]

| Phase Distribution | SSR-PDR | PDR-CDR | CDR-TRR | TRR-FQR |
|---|---|---|---|---|
| Effort | 23 | 29 | 22 | 26 |
| Schedule | 39 | 25 | 15 | 21 |

**SoftCost-Ada.** SoftCost-Ada's allocation of costs to phases of the life-cycle differs from the traditional allocation of time and effort to life-cycle phases [RCI88]. Table 2-4 provides a comparison of SoftCost-R versus SoftCost-Ada phase distribution.

The effort and schedule distribution for SoftCost-Ada are adjusted toward the 40:20:40 allocation if the Ada development process follows a more traditional approach used for other HOLs.

TABLE 2-4

SoftCost-R VS. SoftCost-ADA PHASE DISTRIBUTION

SoftCost-R

| Phase Distribution | SDR-CDR | CDR-TRR | TRR-FQR |
|---|---|---|---|
| Effort | 40 | 20 | 40 |
| Schedule | 45 | 25 | 30 |

SoftCost-Ada

| Phase Distribution | SDR-CDR | CDR-TRR | TRR-FQR |
|---|---|---|---|
| Effort | 50 | 15 | 35 |
| Schedule | 50 | 15 | 35 |

## 2.2.2 Modern Software Development Practices

The software development process encompasses both the wide variety of development paradigms: waterfall development, incremental development, spiral development, pilot development; and the development practices: object oriented design, structured analysis, incremental development, prototyping, program libraries. A combination of structured analysis and object oriented design (OOD) used within a waterfall development paradigm are development processes most frequently associated with Ada in recent efforts. In 1985-1986, TRW developed an initial version of the Ada Process Model that reorients portions of the software development process to capitalize on Ada strengths and to reinforce more efficient software production methods. The Ada Process Model focusses on utilizing the programming-in-the-large features of Ada (primarily, compilable package specifications) to enhance design thoroughness and eliminate major risk items. The Ada Process Model was designed to eliminate inefficiencies of most previous projects "brought on when large numbers of project personnel are working in parallel on tasks which are closely intertwined, incompletely defined, continually changing, and not well prepared for downstream integration [BOEH87]". The primary features of the Ada Process Model include the use of:

- Small up front design teams
- Planned incremental development
- Compilable package specifications by PDR
- Continuous integration via package specifications
- Technical walkthroughs proceeding SSR and PDR.

The Ada Process Model, differs substantially from traditional development practices where main points of interest are the Preliminary Design Review (PDR) and the Critical Design Review (CDR). Interviewees cited in the ESD Report believed that as time progresses, these conventional milestones may no longer be appropriate for Ada software development [TECO87]. Instead they believe that criteria for meeting milestones might coincide with the following points of interest:

- Noting the point in software where all packages and procedures are named.
- Noting when the type statements are identified
- Determining how many objects are identified at all of those times.

Of the cost models reviewed in this study, only SASET and SPQR/20 did not take this issue into account. SYSTEM-3, PRICE S, Ada COCOMO and SoftCost-Ada adjust for the influence of modern software engineering practices to varying degrees, primarily through model input parameters.

**SYSTEM-3.** SYSTEM-3 views the use of modern development practices as an influence to software cost. These influences, however are not considered unique to Ada. The SYSTEM-3 input parameter, Modern Development Practices (MODP), rates the developer's use of modern development practices throughout the project. Factored into the rating for MODP are the development team's previous experience with practices and whether the practices have been used successfully on prior contracts.

**PRICE S.** This issue is accounted for in the productivity factors and schedule multipliers which are calibrated to include the effects of current software engineering practices measured from completed programs. Subsequent application of these factors assumes that conditions present in the calibrated projects continue to hold, thus incorporating the effects of the developers experience with Modern Software Engineering Practices [PARK89].

**Ada COCOMO.** Several input parameters are used to describe the software development process:

- Use of modern programming practices
- Experience with the Ada Process Model
- Design thoroughness at PDR, unit package specs compiled, body outlined
- Risks eliminated at PDR
- Requirements volatility during development.

Most of the factors rate the extent of compliance with the Ada Process Model discussed previously. Use of modern software practices and compliance with the Ada Process Model leads to an overall reduction in project effort.

**SoftCost-Ada.** SoftCost-Ada reflects the modern software engineering practices issue in three of its input parameters and its calibration coefficients:

- Use of Modern Software Methods
- Ada Methodology Experience
- Team Capability.

The Use of Modern Software Methods parameter asks the user to enter the type of software method being used: structured programming, OOD, Ada packaging methods, etc. The second parameter, Ada Methodology Experience, rates the developer's experience with the chosen development methodology. If experience with this methodology is high, then the project will benefit. The last parameter, Team Capability, addresses the issue of the type of teams used in the development. Participatory and interdisciplinary teams are a significant factor in the software being developed by consensus rather than by decree [RCI88].

### 2.2.3 Learning Curve

The learning curve is driven by two issues: the difficulty of learning the language and the amount of time to become proficient in it. Experts cited in the ESD report felt that the learning curve in Ada projects may have more cost impact because of Ada's novelty and complexity [TECO87]. Because Ada is a relatively new language and Ada Programming Support Environments (APSEs) are still evolving, a body of trained personnel are not available to develop new Ada systems. Ada projects will not be able to benefit from lessons learned in previous projects which in turn will lengthen the development schedule. In addition, most respondents felt that Ada requires more experience than other languages before personnel can become proficient. This is because of the unique features of the language (generics, tasks, overload operators, exception handlers) that are not present in other languages. It was generally agreed that it will take longer to appreciate the tradeoffs in determining which feature of the language is best to implement a particular algorithm. Although the time to become proficient in the Ada language seemed longer than other languages, the experts also believed that there would be a "fusion" point where proficiency in the language would become evident and the benefits of the language would be seen in better design, code, and more reusability [TECO87].

All of the cost models studied accounted for the difficulty of learning a language, but the Ada-specific models were the only models that treated the Ada learning curve differently from other languages. For each model, the factor for learning the language was determined by inputs for the experience of the project personnel. With regards to the second issue, the amount of time to become proficient in the Ada language, only Ada-specific models accounted for this issue. Their philosophies are discussed below.

**Ada COCOMO.** Proficiency in Ada versus proficiency in other HOLs can be assessed by comparing the factors associated with the Language Experience (LEXP) input parameter in Table 2-5.

TABLE 2-5

COMPARISON OF STANDARD COCOMO AND ADA COCOMO LANGUAGE EXPERIENCE FACTORS

| Rating Level | Rating Scale | Standard | Ada |
|---|---|---|---|
| VL | <= 1 month | 1.14 | 1.26 |
| L | 4 months | 1.07 | 1.14 |
| NOM | 1 year | 1.00 | 1.04 |
| H | 3 years | 0.95 | 0.95 |
| VH | >= 6 years | 0.95 | 0.86 |

As the chart indicates, larger penalties occur because of Ada ignorance while larger benefits occur because of acquired Ada expertise [TRW88].

**SoftCost-Ada.** A significant finding of RCI researchers was that the exponents used in SoftCost-Ada effort equations did not stabilize until the team assigned to the project had at least 3-5 Ada projects worth of Ada experience. In addition, experience was also influenced by the following factors: analyst capability, applications experience, environment experience, and language and methodology experience. Table 2-6 depicts how the power law changes as a function of the number of Ada projects completed and the other contributing factors [RCI88].

TABLE 2-6

POWER LAW AS A FUNCTION OF THE NUMBER OF ADA PROJECTS COMPLETED

| Number of Completed Ada Projects | P1 | P2 |
|---|---|---|
| 0 | 1.20 | 0.40 |
| 1 | 1.12 | 0.39 |
| 2 | 1.08 | 0.38 |
| 3 | 1.00 | 0.37 |
| 4 | 0.95 | 0.36 |
| 5 | 0.95 | 0.35 |
| 6 | 0.95 | 0.35 |

P1 = Effort Exponent
P2 = Schedule Exponent

### 2.2.4 Reusability Impact

Reusability includes not only developing reusable software but also accounting for components that are reused from other software. It is considered to be a potential cost driver for two separate reasons. First, if software is to be developed for reuse, additional cost impacts will be associated with packaging, increased documentation and stricter reliability requirements. Second, if software from another project is to be reused in a new project, then the development cost should decrease because less software has to be written. However, the costs associated with managing reusable components need to be accounted for because reuse cannot be accommodated without use of such an infrastructure.

Of the cost models examined, all of them account for the second reusability issue, the usage of reusable software. In every model, size and language used for the reusable components were ascertained. Only three models (PRICE S, Ada COCOMO, and SoftCost-Ada) however, took into account the first issue, developing reusable software.

**PRICE S.** PRICE S offers two methods for estimating development of reusable code. First, estimators can describe the increase in requirements to produce reusable code by adjusting the development complexity by +.1 to +.2. This view is mainly used when developers intend to use time, rather than adding people, to meet reusability requirements. Second, estimators can describe reusability as an increase in the application difficulty. Reusable components can be grouped separately, with each group assigned its own application value. This view is appropriate when both resources and time will be used to achieve reusability [PARK89].

TABLE 2-7

ADA COCOMO DEGREE OF REUSE PARAMETER

| Rating | Rating Scale | Ada |
|---|---|---|
| NOM | Not for Reuse Elsewhere | 1.0 |
| H | Reuse Within Single Mission | 1.10 |
| VH | Reuse Across Single Product | 1.30 |
| EH | Reuse in Any Application | 1.50 |

**Ada COCOMO.** Development of reusable code is accounted for in the Degree of Reuse (RUSE) parameter. With this input, the estimator enters the degree of reusability for which the software is being built. The RUSE parameter was incorporated as an improvement to standard COCOMO as well as a feature of Ada COCOMO. Table 2-7 provides the RUSE cost driver ratings and associated effort multipliers.

**SoftCost-Ada.** SoftCost-Ada views development of reusable components in Ada differently than development of reusable components in other languages. Its input parameter, RUSE, is one of the four new and unique cost drivers added to SoftCost-Ada that originally was not found in SoftCost-R [RCI88]. The main philosophy behind this parameter is that Ada has specific features, generics, which have been included in the language to make developing reusable components easier. Further, as explained in section 2.2.3, once the developer becomes more proficient in the language, reusable software will be even easier to develop. The actual determination behind SoftCost-Ada's reuse factor is considered proprietary; therefore, the parameter values cannot be shown.

### 2.2.5 Influence of the Ada Language Features

Ada has several different features that could potentially impact productivity: packages (collections of related programs and data), overload operators (the feature of giving a new meaning to an operator, useful for defining arithmetic operations for types that are not built into Ada), strong typing (the restriction against mixing data types across assignments in expressions), generics (a method of overcoming Ada's sometimes overly strong typing), tasks (a method to allow concurrent processing), and exception handlers (a method to capture program errors and continue processing). The complexity of the features to be used on a given system depends strongly on the application type of that system (avionics, business, command and control) [TECO87]. For example, a business system and a command and control system would both contain exception handlers but the difference would occur in the complexity of the exception handler. In a business system, the exception handlers may be less complex because a critical error would cause only loss of data. In a command and control system, the handlers would be more complex because a critical error could cause loss of life.

Non-Ada-specific models do not differentiate between specific Ada language features that are utilized within a program. The complexity of the application is taken into account, but there are no benefits when using Ada language features extensively. For example, packaging technology may be utilized to varying degrees on different projects. Non-Ada-specific models do not provide a direct means to evaluate the Ada packaging technology aspect. With regards to the Ada-specific models, Ada features such as packaging were accounted for in the factors that reflect the extent of compliance with the Ada Process Model and in the complexity parameters. A comparison of the Software Product Complexity factor in Standard COCOMO versus Ada COCOMO in

Table 2-8 illustrates that Ada features make many complex constructs more straightforward [BOEH86].

TABLE 2-8

COMPARISON OF SOFTWARE PRODUCT COMPLEXITY IN STANDARD COCOMO VERSUS ADA COCOMO

| Rating | Standard | Ada |
|---|---|---|
| VL | 0.70 | 0.73 |
| L | 0.85 | 0.85 |
| NOM | 1.00 | 0.97 |
| H | 1.15 | 1.08 |
| VH | 1.30 | 1.22 |
| EH | 1.65 | 1.43 |

### 2.2.6 Influence of the Ada Programming Support Environment

The Ada Programming Support Environment is a set of coordinated tools for the Ada language. It contains such software tools as [BOOC87]:

- Text editor
- Pretty printer
- Compiler
- Linker
- Set-use static analyzer
- Control-flow static analyzer
- Dynamic analysis tools
- Terminal interface routines
- File administrator
- Command interpreter
- Configuration manager
- APSE interface to underlying machine.

Although the APSE has not matured to its fullest potential, experts cited in the ESD Report said that once it does, the cost of software development will decrease [TECO87]. These tools will enable the developer to effectively and efficiently use the Ada language resulting in increased software productivity.

Every model in the test-case study accounted for the APSE as a cost driver and instructions are provided in each model to help determine the recommended input for the tools environment. Each model rewards the developer for upgrades in the tool support. The SoftCost-Ada model was recalibrated to reflect the impact of a workstation/APSE strategy. In a study conducted over three years, RCI developed new calibrations to show the effects of APSE factors on productivity and cost [RCI89].

### 2.2.7 Language and Tool Maturity

Some of the language and tool maturity issues have been discussed in previous sections (2.2.3 and 2.2.6), but one topic still exists, optimization. The optimization problem can be illustrated in examining Ada tasking. Tasks are used as a mechanism for development of concurrent software. Recently, this feature has come under a lot of fire at Ada Jovial User Group (AdaJUG) meetings because Ada lacks a suitable method of providing regularly scheduled tasks within maximum time constraints [TECO87]. Currently, tasks do not meet embedded (airborne) system needs. This is because of large object code sizes, slow run times, and processors with limited memory and severe time constraints [TECO87]. All of these factors are part of the optimization problem. Eventually, compiler writers should solve the optimization problem, but currently optimization issues have an impact on the software development.

Although the optimization issue is prominent in today's software, only SoftCost-Ada handles this issue through its degree of real-time parameter. This questions essentially asks "How much Ada tasking is involved?" Other cost models exclude the issue of tasking from their equations with the assumption that as technology improves, optimization, in terms of tasking, will no long be an issue.

### 2.2.8 Hawthorne Effect

The Hawthorne Effect encompasses the philosophy that experimental Ada projects may be unduly influenced by the "better" personnel who are allocated to the project and the fact that the project is being closely monitored [TECO87]. Respondents from the ESD report thought that this issue could be a potential cost driver.

A recent article [ORME86] describes an investigation in which 12 sites were asked how personnel were selected for Ada projects. More than 300 staff were allocated in a ratio of about 60:40, conscript:voluntary. There was no evidence that Ada projects were given an undue priority for resources. Project managers always fought for the best staff available. In conclusion, there was no evidence that either supported or disproved the idea of the Hawthorne effect.

None of the models reviewed consider the Hawthorne effect.

# 3.0 APPLICATION OF SELECTED COST MODELS TO ADA PROJECTS

## 3.1 BACKGROUND

This section provides an overview of the Ada projects targeted in the model applications. The methodology used to collect, validate and normalize project data, derive model inputs, and, subsequently, interpret the results is presented.

## 3.2 OVERVIEW OF ADA PROJECTS TARGETED IN THE TEST CASE STUDY

Data was aquired for 10 Ada projects; however, only eight were targeted in the test case study. Table 3-1 provides a summary of the application and functionality of projects. In addition, Appendix H presents an overview of each project with regard to the following areas:

- Product Description
- Software Size
- Development Process
- Computer System.

These areas map to model input categorizations contained in Appendix B.

## 3.3 METHODOLOGY

Project data needed to be validated and normalized before it could be used for input to the models. The following subsections correspond to activities that supported the model applications:

- Data Collection
- Data Validation

TABLE 3-1

PROJECT FUNCTIONAL DESCRIPTIONS

| PROJECT NUMBER | DESCRIPTION |
|---|---|
| 1 | Commercial real-time control center traffic planning and monitoring system that provides traffic planning, controller interfaces, and traffic supervision. |
| 2 | Tactical command and control system that supports planning, directing and executing of orders issued by the commander. |
| 3 | APSE interface tool that provides multiple screen images and runs a host operation system command interpreter in each window. |
| 4 | Ada interactive environment that improves productivity for the development of large Ada software systems. Provides interactive Ada compilation and cross compiler, debugging tools, and software management. |
| 5 | Development tool that integrates design or development of hardware and software for embedded systems before prototyping. |
| 6 | Avionics guidance and control system with software for pre-programmed flight paths and full up, guided flight. |
| 7 | Command and control system that is designed to support testing and evaluation of tactical doctrine. |
| 8 | Command and control system that performs target detection, target recognition and identification, and laser target designation for a mission payload subsystem. |
| 9 | Component within an avionics system that controls stabilizer position, aileron lockout, and ratio of rudder movement. All firmware. |
| 10 | Command and control system that will serve as both a fire support control and coordination system and a field artillery system. |

- Normalization
- Parameter Ratings Selection.

### 3.3.1 Data Collection

Projects targeted in this test case study were obtained from two primary sources:

1. Air Force Electronic Systems Division (ESD) at Hanscom AFB.
2. Direct inquiry of organizations identified in the AJPO Ada Usage Database.

Four projects (identified as Projects 2, 3, 4, and 10 in Table 3-1) were obtained from ESD. One of these projects (Project 10) was discarded because effort expenditure data was omitted from the data collection form.

Inquiries into the projects listed in the AJPO Ada Usage Database resulted in 19 respondents who agreed to provide data for this study. Of these newly identified projects, four were eventually able to provide requested data in a timely manner. These were Projects 5, 6, 7, and 8 in Table 3-1.

The two remaining projects targeted in the study, Projects 1 and 9, were obtained through random inquiries. One of these projects (Project 9) was discarded because of its size (< 5000 SLOC).

Table 3-2 provides an overview of projects that were targeted in the test case study.

TABLE 3-2

PROJECT SUMMARY INFORMATION*

| PROJECT NUMBER | ACTUAL EFFORT (PM) | TYPE OF PROJECT | TYPE OF CONTRACT | PROJECT SIZE** (SLOC) | Ada EXPERIENCE (YEARS) | DEVELOPMENT APPROACH |
|---|---|---|---|---|---|---|
| 1 | 302 | Command & Control | Commercial | 50,000 | 1 | Object Oriented Design |
| 2 | 684 | Command & Control | Government | 115,000 | 1 | Object Oriented Design |
| 3 | 134 | Tool/ Environment | Commercial | 18,640 | 1 | Object Oriented Design |
| 4 | 692 | Tool/ Environment | Commercial | 480,000 | 5 | Object Oriented Design |
| 5 | 144 | Tool/ Environment | Commercial | 136,000 | 3 | Object Oriented Design |
| 6 | 190 | Avionics | Government | 31,800 | 0 | Structured Design |
| 7 | 696 | Command & Control | Government | 69,160 | 0 | Structured Design |
| 8 | 322 | Command & Control | Government | 18,300 | 0 | Object Oriented Design |

* Projects 9 and 10 were discarded
** Size includes reused and non-Ada code. See Appendix H for a software size description of each project.

3.3.2 Data Validation

Data was provided for each Ada project in the form of completed project questionnaires. The first set of questionnaires contained data collected by Tecolote Research on four projects using data collection forms prepared for USAF/ESD. The second set of questionnaires contained data collected by IIT Research Institute (IITRI) using forms IITRI devised from an examination of model input parameters. IITRI's forms are an extension of RCI's Data Collection Form (RCI-TP-0197-FINT).

The new forms were developed in view of feedback from software developers providing the project information who pointed out the ESD forms were intimidating. Validating the data contained within the forms was difficult because of the sheer volume of information contained within them. Further, much of the detailed information requested on the forms was not needed to run the models. Hence, significant factors that impact a project's schedule were not readily apparent.

Inconsistencies or discrepancies with regard to the contents of the project questionnaire were noted and later resolved through subsequent conversations with the software developer providing project data. Separate validation was also conducted for the sensitivity analysis portion of this study [REIF89]. In several instances some changes to the ESD data were recommended regarding quality of staffing, tools, type of teams used, and average person-months based on an examination of other sources. Personnel who completed the forms were contacted to verify responses on the project questionnaires. **Changes to the project data which were recommended by personnel who validated the data were implemented only if those who collected the data agreed with the recommendations.**

## 3.3.3 Normalization

In order to use the data for subsequent analysis, project data needed to be in a comparable format. Each software developing organization that submitted project data defined key productivity parameters differently. The following definitions were employed to normalize SLOC, person months, and scope of effort across projects:

- An Ada source line of code was defined using terminal semicolons.
- A person month was defined to be 160 hours of direct chargeable labor.
- Given the typical development cycle during full-scale development shown in Figure 3-1, the scope of the effort provided by the developers extended from the System Design Review (SDR) to final software acceptance. System requirements analysis and system integration and testing were not included in the base effort.
- Software activities included in the effort provided by each developer did not include Quality Assurance (QA), Configuration Management (CM), and project level management functions for all projects targeted in the study. Table 3-3 shows the software cost elements that are assumed to be covered by actual effort provided by each software developer providing project data, in the terminology of each model.

### Ada Source Lines of Code (ASLOC)

Five definitions are currently being advanced for an ASLOC [RCI88]:

1. Physical Lines - any carriage return or line feed including comments and blank lines. Reusable code is counted the first time it is instantiated.
2. Non-Comment, Non-Blank Lines - physical lines excluding comments and blank lines.

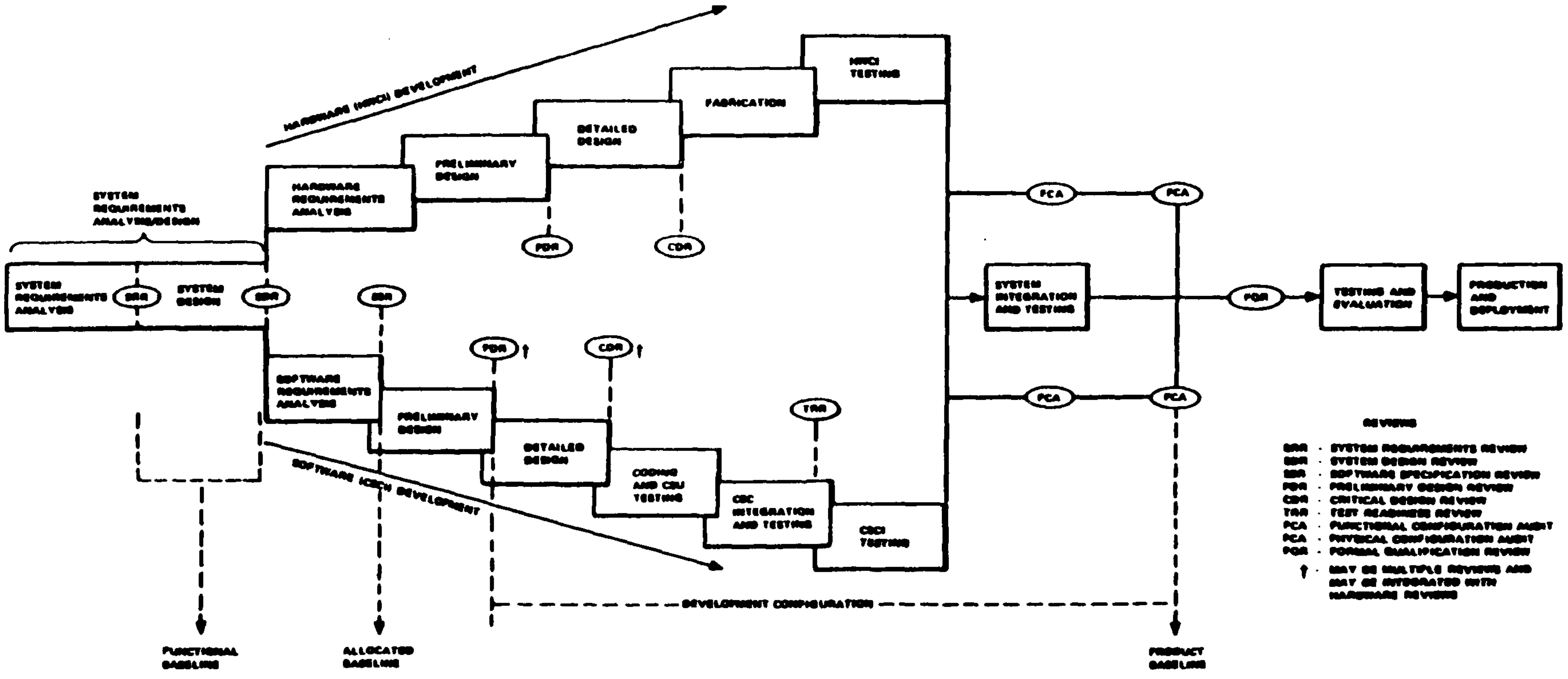
SYSTEM REQUIREMENTS ANALYSIS/DESIGN
SYSTEM REQUIREMENTS ANALYSIS
SRR
SYSTEM DESIGN
SDR
HARDWARE (HWCI) DEVELOPMENT
HARDWARE REQUIREMENTS ANALYSIS
PRELIMINARY DESIGN
DETAILED DESIGN
FABRICATION
HWCI TESTING
PDR
CDR
SSR
PDR †
CDR †
TRR
SOFTWARE REQUIREMENTS ANALYSIS
PRELIMINARY DESIGN
DETAILED DESIGN
CODING AND CSU TESTING
CSC INTEGRATION AND TESTING
CSCI TESTING
SOFTWARE (CSCI) DEVELOPMENT
FCA
PCA
SYSTEM INTEGRATION AND TESTING
FQR
TESTING AND EVALUATION
PRODUCTION AND DEPLOYMENT
FUNCTIONAL BASELINE
ALLOCATED BASELINE
DEVELOPMENT CONFIGURATION
PRODUCT BASELINE
REVIEWS
SRR - SYSTEM REQUIREMENTS REVIEW
SDR - SYSTEM DESIGN REVIEW
SSR - SOFTWARE SPECIFICATION REVIEW
PDR - PRELIMINARY DESIGN REVIEW
CDR - CRITICAL DESIGN REVIEW
TRR - TEST READINESS REVIEW
FCA - FUNCTIONAL CONFIGURATION AUDIT
PCA - PHYSICAL CONFIGURATION AUDIT
FQR - FORMAL QUALIFICATION REVIEW
† - MAY BE MULTIPLE REVIEWS AND MAY BE INTEGRATED WITH HARDWARE REVIEWS

Figure 3-1. Software Development Cycle for Full-Scale Development [DoD-STD-2167A]

TABLE 3-3

SOFTWARE ACTIVITIES INCLUDED IN THE ACTUAL EFFORT PROVIDED BY EACH SOFTWARE DEVELOPER PROVIDING PROJECT DATA IN THE TERMINOLOGY OF EACH MODEL

| | | | |
|---|---|---|---|
| | **COSTMODL** | | **PRICE S** |
| 100% | Requirements Analysis | 100% | Software Design |
| 100% | Product Design | 100% | Programming |
| 100% | Programming | 100% | Documentation |
| 100% | Test Planning | * | Systems Engineering and Program Management |
| 100% | Verification and Validation | 0 | QA |
| 0% | Project Office | 0 | CM |
| 100% | Manuals | | |
| | **SASET** | | **SoftCost-Ada** |
| 100% | Software Engineering | 100% | Software Development |
| 100% | Systems Engineering | 0 | Software Management |
| 0 | QA | 0 | Software CM |
| 100% | Test Engineering | 0 | Software Quality Evaluation |
| | **SPQR/20** | | **SYSTEM-3** |
| 0 | Planning | 100% | Systems Engineering |
| 100% | Requirements | 0 | Project Management |
| 100% | Design | 100% | Design |
| 100% | Coding | 100% | Programmers |
| 100% | Integration/Test | 0 | QA |
| 100% | Documentation | 0 | CM |
| 0 | Management | 100% | Test |
| | | 100% | Data Manipulation |

* Note: The following conversion was performed to discount Project Management from the total estimate [PARK89A]:

Systems Engineering (MM) =

((Design Cost / .69) * 31%) + ((Coding Cost / .75) * 25%)

3. Terminal Semicolons - a statement terminated by a semicolon, including data declarations, code used to instantiate a reusable component and the reusable component itself the first time it was instantiated. When multiple semicolons are used within a declaration statement, the terminating semicolon is used to define the termination of the source line of code. For example, a package specification which included a statement that spans ten lines and is terminated by a single semicolon would count as one ASLOC. Comments, blank lines and non-deliverable code are not included in the line count.

4. Essential or Limited Terminal Semicolons - terminal semicolons excluding those used in data declarations or formal parameter lists.

5. Body Semicolons - a statement terminated by a carriage return in the specification and a terminal semicolon in the body of an Ada program, including data declarations and code used to instantiate a reusable component itself the first time it was instantiated. Comments, blank lines and non-deliverable code are not included in the line count.

An ASLOC was defined using terminal semicolons. Counts provided by project developers using other definitions were converted using the following conversion factors [RCI88]:

| Definition | Conversion Factor |
|---|---|
| • Non-Comment, Non-Blank Lines | Reduce count by 20% |
| • Terminal Semicolons | 1 to 1 |
| • Essential Semicolons | Increase count by 30% |
| • Body Semicolons | Reduce count by 5% |

One of the projects (Project 5) provided a size estimate in physical lines. After a discussion with the developer of Project 5, it was decided to run two estimates: one assuming that 1 of every 5 lines was commented and the second reflecting that 1 in 10 lines was commented. Line counts were adjusted to account for commented lines. The adjusted line count was reduced by 20% to normalize for carriage returns.

## 3.3.4 Criteria For Ratings Selection

For all projects, model inputs were set at an average rating if there was no data to support deviating from the nominal value. There were instances, however, when a model input parameter had no associated nominal rating or information provided on the forms was disregarded. One instance was with regard to the project's required development schedule. For consistency, this factor was assumed to be nominal regardless of the developer's assessment. Other criteria pertain to specific models.

### PRICE S

**Schedule.** The schedule input for PRICE S requires that the System Design Review (SDR) or Software Requirements Review (SRR) be specified. These review dates were not provided for any of the projects targeted in the study. Therefore, it was assumed that the date of SRR would fall on the date which corresponded to one fourth of the entire development schedule.

**Productivity Factor.** The productivity factor is an empirically derived value that describes the demonstrated performance of each developing organization. This value was assumed to be nominal.

**Complexity.** The complexity adjustment for a new language was assumed +.1 since the this factor could range from 0.0 to 0.2.

**Source Code Mix.** Non-ESD project data did not contain a breakdown of the software profile. The mix element application value for these projects was assumed to be the value associated with the application type of the software.

### SoftCost-Ada

**Ada Projects Completed.** The response to the number of Ada projects completed by the development team was incremented by one if an Ada PDL was used during the development.

**Software Organizations Involved.** Projects 2, 3, and 4 did not supply the number of organizations providing level of effort support (i.e. CM and QA). Development contractors were assumed to have three software organizations.

## 3.4 RESULTS

Section 3.4.1 presents the results of the model applications that required some normalization. Tables are provided to illustrate the relative difference between estimated and actual effort for each project. Sections 3.4.2 and 3.4.3 compare the performances of models in the areas of accuracy, consistency, contract type, application type, and Ada issues.

### 3.4.1 Normalization

Model outputs were subjected to a normalization process so that results would reflect the same scope of coverage. This process consisted of subtracting model efforts that were associated with phases not included in the actual effort (effort that was prior to SDR and subsequent to the software acceptance review). In addition, Quality Assurance, Configuration Management, and Program Management efforts were also subtracted because these organizations were not part of the effort provided (Refer to Section 3.3.3, Normalization). Table 3-4 is provided to illustrate the normalization process that was applied to each model.

TABLE 3-4

NORMALIZATION OF PROJECT 1 EFFORT

MODEL: COSTMODL

| | MM | Normalization | Normalized MM |
|---|---|---|---|
| Requirements Analysis | 17.8 | 100% | 17.8 |
| Product Design | 33.1 | 100% | 33.1 |
| Programming | 93.4 | 100% | 93.4 |
| Test Planning | 13.2 | 100% | 13.2 |
| Verification & Valid | 31.6 | 100% | 31.6 |
| Project Office | 19.7 | 0% | 0 |
| CM/QA | 15.3 | 0% | 0 |
| Manuals | 16.1 | 100% | 16.1 |
| TOTAL EFFORT: | 240.2 | NORMALIZED EFFORT: | 205.2 |

TABLE 3-4 (Continued)

MODEL: SoftCost-Ada

| | MM | Normalization | Normalized MM |
|---|---|---|---|
| SW Development | 319 | 100% | 319 |
| SW Management | 34.6 | 0% | 0 |
| SW Configuration Mgt | 17.3 | 0% | 0 |
| SW Quality Evaluation | 20.8 | 0% | 0 |
| | ----- | | ---- |
| TOTAL EFFORT: | 391.7 | NORMALIZED EFFORT: | 319 |

MODEL: PRICE S

| | MM | Normalization | Normalized MM |
|---|---|---|---|
| SW Design | 410 | 100% | 410 |
| Programming | 245 | 100% | 245 |
| Documentation | 86 | 100% | 86 |
| Sys Eng./Program Mgt | 221 | * | 47.29 |
| Quality Assurance | 92 | 0% | 0 |
| Configuration Mgt | 98 | 0% | 0 |
| | ---- | | ---- |
| TOTAL EFFORT: | 1152 | NORMALIZED EFFORT: | 788.29 |

* NOTE: Normalization is performed by the following conversion:

(design cost/0.69 * 31%) + (coding cost/.75 * 25%)

MODEL: SPQR/20

| | MM | Normalization | Normalized MM |
|---|---|---|---|
| Planning | 2.68 | 0% | 0 |
| Requirements | 14.92 | 100% | 14.92 |
| Design | 73.25 | 100% | 73.25 |
| Coding | 43.50 | 100% | 43.50 |
| Integration/Test | 42.31 | 100% | 42.31 |
| Documentation | 62.88 | 100% | 62.80 |
| Management | 32.38 | 0% | 0 |
| | ----- | | ---- |
| TOTAL EFFORT: | 271.91 | NORMALIZED EFFORT: | 236.78 |

TABLE 3-4 (Continued)

MODEL: SYSTEM-3

| | MM | Normalization | Normalized MM |
|---|---|---|---|
| System Engineering | 48 | 100% | 48 |
| Project Management | 35.7 | 0% | 0 |
| Design | 141.8 | 100% | 141.8 |
| Programmers | 148.8 | 100% | 148.8 |
| Quality Assurance | 25.1 | 0% | 0 |
| Configuration Mgt | 23.3 | 0% | 0 |
| Test | 110.3 | 100% | 110.3 |
| Data Manipulation | 11.5 | 100% | 11.5 |
| | ---- | | ---- |
| TOTAL EFFORT: | 544.5 | NORMALIZED EFFORT: | 460.4 |

MODEL: SASET

| | MM | Normalization | Normalized MM |
|---|---|---|---|
| Software Engineering | 307.59 | 100% | 307.59 |
| Systems Engineering | 53.76 | 100% | 53.76 |
| Q.A. | 23.31 | 0% | 0.00 |
| Test Engineering | 62.18 | 100% | 62.18 |
| | ----- | | ----- |
| TOTAL EFFORT: | 446.84 | NORMALIZED EFFORT: | 423.53 |

After normalization, model estimates were compared to actual effort data to determine the relative error.

In three instances, models were not used to derive project estimates. The COSTMODL implementation of IOC Ada COCOMO provides equations that apply to the COCOMO embedded mode of software development. The model co-developer, Dr. Barry Boehm, confirmed that equations currently do not exist for the organic and semidetached modes. Two of the projects targeted in the test case study, Projects 3 and 4, fall into the semidetached category of software development. Thus, COSTMODL was not applied to these projects.

Project 4 was used by RCI for the development of the SoftCost-Ada model. Therefore, so as not to compromise credibility of the test case study results, Project 4 was not targeted for application to SoftCost-Ada.

### 3.4.2 Comparison of Model Results

As stated in the objective, model performances were analyzed with regard to five areas: overall accuracy, overall consistency, contract type, application type, and Ada issues. Model performances varied with regard to each assessment criteria. The following paragraphs specify the findings for each of these areas.

#### 3.4.2.1 Overall Accuracy

Model effort and schedule projections were compared to actual effort and schedule expended by the project developer to ascertain which models provided the most accurate results. Results in view of projected effort versus actual effort are provided in Tables 3-5 and 3-6. A comparison of projected schedules versus actual schedules are illustrated in Tables 3-7 and 3-8. The number of effort estimates that were accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SoftCost-Ada | 4 out of 7 | 0% to 13% |
| 2. | SASET | 4 out of 8 | -29% to 29% |
| 3. | SPQR/20 | 3 out of 8 | -22% to 19% |
| 4. | COSTMODL | 2 out of 6 | -25% to -1% |
| 5. | PRICE S | 0 out of 8 | Not Applicable |
| 6. | SYSTEM-3 | 0 out of 8 | Not Applicable |

TABLE 3-5

MODEL EFFORT PROJECTIONS

| Number | Actual Effort | SoftCost-Ada | COSTMODL | PRICE S | SASET | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|---|
| 1 | 302 | 319 | 205 | 788 | 424 | 460 | 237 |
| 2 | 684 | 1446 | 516 | 1487 | 635 | 2258 | 811 |
| 3 | 134 | 137 | Not Run | 42 | 95 | 90 | 296 |
| 4 | 692 | Not Run | Not Run | 2223 | 1931 | 2323 | 2876 |
| 5 | 144 | 144 | 356 | 985 | 725 | 1366 | 260 |
| 6 | 190 | 360 | 93 | 879 | 335 | 641 | 109 |
| 7 | 696 | 1621 | 692 | 1827 | 822 | 2303 | 560 |
| 8 | 322 | 364 | 108 | 623 | 416 | 728 | 86 |

TABLE 3-6

RELATIVE ERRORS FOR MODEL EFFORTS WHEN COMPARED TO ACTUAL EFFORTS

| Number | Actual Effort | SoftCost-Ada | COSTMODL | PRICE S | SASET | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|---|
| 1 | 302 | 6% | -32% | 161% | 40% | 52% | -22% |
| 2 | 684 | 111% | -25% | 117% | - 7% | 230% | 19% |
| 3 | 134 | 2% | Not Run | -69% | -29% | -33% | 121% |
| 4 | 692 | Not Run | Not Run | 221% | 179% | 236% | 316% |
| 5 | 144 | 0% | 147% | 584% | 403% | 849% | 81% |
| 6 | 190 | 89% | -51% | 363% | 76% | 237% | -43% |
| 7 | 696 | 133% | - 1% | 163% | 18% | 231% | -20% |
| 8 | 322 | 13% | -66% | 93% | 29% | 126% | -73% |

TABLE 3-7

MODEL SCHEDULE PROJECTIONS

| Number | Actual Schedule | SoftCost-Ada | COSTMODL | PRICE S | SASET | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|---|
| 1 | 27 | 26.04 | 20.3 | 50.0 | 28.0 | 25.12 | 33.9 |
| 2 | 38 | 61.48 | 27.4 | 45.0 | 28.7 | 31.45 | 57.6 |
| 3 | 35 | 17.39 | Not Run | 20.0 | 15.1 | 13.34 | 57.3 |
| 4 | 49 | Not Run | Not Run | 29.0 | 51.9 | 43.11 | 65.6 |
| 5 | 12 | 16.54 | 23.5 | 46.0 | 38.2 | 36.34 | 38.7 |
| 6 | 40 | 29.52 | 14.3 | 47.0 | 18.0 | 29.09 | 33.6 |
| 7 | 62 | 85.16 | 28.5 | 90.0 | 30.4 | 41.90 | 56.4 |
| 8 | 56 | 25.66 | 16.0 | 58.0 | 22.4 | 28.19 | 34.6 |

TABLE 3-8

RELATIVE ERRORS FOR MODEL SCHEDULES WHEN COMPARED TO ACTUAL SCHEDULES

| Proj. Number | Actual Schedule | SoftCost-Ada | COSTMODL | PRICE S | SASET | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|---|
| 1 | 27 | - 4% | -25% | 85% | 4% | -7% | 26% |
| 2 | 38 | 62% | -28% | 18% | -24% | -17% | 52% |
| 3 | 35 | -50% | NR | -43% | -57% | -62% | 64% |
| 4 | 49 | NR | NR | -41% | 6% | -12% | 34% |
| 5 | 12 | 38% | 95% | 283% | 218% | 203% | 223% |
| 6 | 40 | -26% | -64% | 18% | -55% | -27% | -16% |
| 7 | 62 | 37% | -54% | 45% | -51% | -32% | - 9% |
| 8 | 56 | -54% | -71% | 3% | -60% | -50% | -38% |

The number of schedule estimates that were accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SYSTEM-3 | 4 out of 8 | -27% to -7% |
| 2. | PRICE S | 3 out of 8 | 3% to 18% |
| 3. | SASET | 3 out of 8 | -24% to 6% |
| 4. | SPQR/20 | 3 out of 8 | -16% to 26% |
| 5. | COSTMODL | 2 out of 6 | -28% to -25% |
| 6. | SoftCost-Ada | 2 out of 7 | -26% to -4% |

### 3.4.2.2 Overall Consistency

Since the objective of the study was to compare model performance when applied to the same project data, special emphasis was placed on obtaining consistent interpretations of project data across all models. To accomplish this task, one person was chosen to derive all inputs for all projects. This insured that the same knowledge base was used to interpret project information and model parameters when inputs were derived. After results were obtained, an analysis of model efforts was performed to establish if results were consistently high or consistently low, eliminating differences between the perspectives of the person deriving the inputs and the model developer. This process involved the following steps:

1. A percentage of actual effort to model effort was calculated.
2. The two extremes were discarded to achieve a truer sampling of percentages.
3. A mean value of the remaining percentages was computed and applied to the given model's estimates.
4. The relative error for each project was recalculated using the adjusted efforts.

The results of this process when applied to each model are illustrated in Tables 3-9 and 3-10. After application of the means, the number of adjusted effort estimates within +-30% were as follows:

TABLE 3-9

RELATIVE ERRORS FOR EFFORT AFTER APPLYING MEANS

| PROJECT NUMBER | SoftCost-Ada Mean=0.87 | COSTMODL Mean=1.46 | PRICE S Mean=0.38 | SASET Mean=0.72 | SYSTEM-3 Mean=0.38 | SPQR/20 Mean=1.02 |
|---|---|---|---|---|---|---|
| 1 | - 8% | - 1% | - 1% | 1% | - 42% | - 20% |
| 2 | 84% | 10% | - 17% | - 33% | 25% | 21% |
| 3 | - 11% | Not Run | - 88% | - 49% | - 74% | 125% |
| 4 | Not Run | Not Run | 22% | 101% | 28% | 324% |
| 5 | - 13% | 261% | 160% | 263% | 260% | 84% |
| 6 | 65% | - 29% | 76% | 27% | 28% | - 41% |
| 7 | 103% | 45% | 0% | - 15% | 26% | - 18% |
| 8 | - 2% | - 51% | - 26% | - 7% | - 14% | - 73% |

TABLE 3-10

RELATIVE ERRORS FOR SCHEDULE AFTER APPLYING MEANS

| PROJECT NUMBER | SoftCost-Ada Mean=0.90 | COSTMODL Mean=1.93 | PRICE S Mean=0.69 | SASET Mean=1.63 | SYSTEM-3 Mean=1.38 | SPQR/20 Mean=0.85 |
|---|---|---|---|---|---|---|
| 1 | -13% | 45% | 28% | 69% | 28% | 7% |
| 2 | 46% | 39% | - 18% | 23% | 14% | 29% |
| 3 | - 55% | Not Run | - 61% | - 30% | - 47% | 39% |
| 4 | Not Run | Not Run | - 59% | 73% | 21% | 14% |
| 5 | 24% | 278% | 165% | 419% | 318% | 174% |
| 6 | - 34% | - 31% | - 19% | - 27% | 0% | - 29% |
| 7 | 24% | - 11% | 0% | - 20% | 7% | - 23% |
| 8 | - 59% | - 49% | - 29% | - 35% | - 31% | - 47% |

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SYSTEM-3 | 5 out of 8 | -14% to 28% |
| 2. PRICE S | 5 out of 8 | -26% to 22% |
| 3. SoftCost-Ada | 4 out of 7 | -13% to -2% |
| 4. COSTMODL | 3 out of 6 | -29% to 10% |
| 5. SASET | 4 out of 8 | -15% to 27% |
| 6. SPQR/20 | 3 out of 8 | -20% to 21% |

After application of the means, the number of adjusted schedule estimates within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SYSTEM-3 | 5 out of 8 | 0% to 28% |
| 2. PRICE S | 5 out of 8 | -29% to 28% |
| 3. SPQR/20 | 5 out of 8 | -29% to 29% |
| 4. SASET | 4 out of 8 | -30% to 23% |
| 5. SoftCost-Ada | 3 out of 7 | -13% to 24% |
| 6. COSTMODL | 1 out of 6 | -11% |

### 3.4.2.3 Analysis By Application Type, Contract Type, and Ada Issues

The last three areas in which model performances were evaluated consisted of the following: contract type, application type, and Ada issues. Table 3-2 provides an overview of projects with regard to these evaluation criteria. Performance in view of remaining criteria was evaluated by comparing accuracy and consistency of effort estimates. Findings are noted below.

**Government Versus Commercial Contracts.** As illustrated in Table 3-2, four government contracts and four commercial contracts were targeted in the test case study. Examination of model performances identified the following trends:

GOVERNMENT CONTRACTS

**Model Accuracy.** SASET provided more accurate results for government contracts where three of four estimates were within 29% of the actual effort. The number of effort estimates accurate with ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SASET | 3 out of 4 | - 7% to 29% |
| 2. COSTMODL | 2 out of 4 | -25% to -1% |
| 3. SPQR/20 | 2 out of 4 | -20% to 19% |
| 4. SoftCost-Ada | 1 out of 4 | 13% |
| 5. PRICE S | 0 out of 4 | Not Applicable |
| 6. SYSTEM-3 | 0 out of 4 | Not Applicable |

**Model Consistency.** SYSTEM-3 provided the most accurate results for government contracts with four of four estimates accurate within 28% after a mean was applied. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SYSTEM-3 | 4 out of 4 | -14% to 28% |
| 2. PRICE S | 3 out of 4 | -26% to 0% |
| 3. SASET | 3 out of 4 | -15% to 27% |
| 4. SPQR/20 | 2 out of 4 | -18% to 21% |
| 5. COSTMODL | 2 out of 4 | -29% to 10% |
| 6. SoftCost-Ada | 1 out of 4 | -2% |

COMMERCIAL CONTRACTS

**Model Accuracy.** SoftCost-Ada was extremely accurate for estimating commercial contracts. Three of three estimates were within 6% of actual effort. The number of effort estimates accurate with ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 3 out of 3 | 0% to 6% |
| 2. SPQR/20 | 1 out of 4 | -22 |
| 3. SASET | 1 out of 4 | -29 |
| 4. COSTMODL | 0 out of 2 | Not Applicable |
| 5. PRICE S | 0 out of 4 | Not Applicable |
| 6. SYSTEM-3 | 0 out of 4 | Not Applicable |

**Model Consistency.** Softcost-Ada provided the most accurate results for commercial contracts with three of three estimates within 13% of actual effort after a mean was applied. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 3 out of 3 | -13% to -8% |
| 2. PRICE S | 2 out of 4 | - 1% to 22% |
| 3. COSTMODL | 1 out of 2 | - 1% |
| 4. SASET | 1 out of 4 | 1% |
| 5. SPQR/20 | 1 out of 4 | -20% |
| 6. SYSTEM-3 | 1 out of 4 | 28% |

**Application Type.** Project data provided by software developers consisted of three different types of applications: command and control (four projects), tools/environment (three projects), and avionics (one project). Performances of models with regard to these application areas were analyzed. Results were inconclusive with regard to the avionics application type because there was only one project of this type targeted in the study. Results are as follows:

COMMAND AND CONTROL

**Model Accuracy.** Estimates generated by SASET and SPQR/20 were most comparable to the actual effort with three of four estimates within 30% of actual effort for command and control type projects. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SASET | 3 out of 4 | - 7% to 29% |
| 2. SPQR/20 | 3 out of 4 | -22% to 19% |
| 3. SoftCost-Ada | 2 out of 4 | 6% to 13% |
| 4. COSTMODL | 2 out of 4 | -25% to -1% |
| 5. PRICE S | 0 out of 4 | Not Applicable |
| 6. SYSTEM-3 | 0 out of 4 | Not Applicable |

**Model Consistency.** Although all of the models did very well, PRICE S provided more accurate results with four out of four estimates within 26% of the actual effort after a mean was applied. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. PRICE S | 4 out of 4 | -26% to 0% |
| 2. SASET | 3 out of 4 | -15% to 1% |
| 3. SYSTEM-3 | 3 out of 4 | -14% to 26% |
| 4. SPQR/20 | 3 out of 4 | -20% to 21% |
| 5. SoftCost-Ada | 2 out of 4 | - 8% to -2% |
| 6. COSTMODL | 2 out of 4 | - 1% to 10% |

TOOLS/ENVIRONMENT

**Model Accuracy.** SoftCost-Ada was accurate within 2% on two out of the two tools/environment projects to which it was applied. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 2 out of 2 | 0% to 2% |
| 2. SASET | 1 out of 3 | -29% |
| 3. COSTMODL | 0 out of 1 | Not Applicable |
| 4. PRICE S | 0 out of 3 | Not Applicable |
| 5. SYSTEM-3 | 0 out of 3 | Not Applicable |
| 6. SPQR/20 | 0 out of 3 | Not Applicable |

**Model Consistency.** SoftCost-Ada also provided the most accurate results in this category with two out of two estimates within 13% of the actual effort after a mean was applied. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 2 out of 2 | -13% to -11% |
| 2. SYSTEM-3 | 1 out of 3 | 28% |
| 3. PRICE S | 1 out of 3 | 22% |
| 4. COSTMODL | 0 out of 1 | Not Applicable |
| 5. SASET | 0 out of 3 | Not Applicable |
| 6. SPQR/20 | 0 out of 3 | Not Applicable |

**Comparison of Ada Issues.** Model philosophies regarding Ada issues were examined to identify any possible reasons for differences between model estimates. An evaluation of project data in light of Ada issues described in Section 2.2 showed that six of the eight issues could not be evaluated for either of two reasons:

1. Project data was not provided that could be used to evaluate the issue.
2. When information was provided, differences in project data with regard to the issue were not apparent.

Three issues were excluded from evaluation because of a lack of data: phase distribution of effort, language and tool maturity, and the Hawthorne effect. Issues excluded because of a lack of definitive data were the reusability impact, influence of Ada language features, and influence of the APSE. The analysis of the remaining issues, modern software development practices and the learning curve, was performed by evaluating model performances based on the design approach and Ada experience of the project developers.

**Modern Software Development Practices: Design Approach.** Of the eight projects targeted in the test case study, two (Projects 6 and 7) utilized a structured design approach, and six used an OOD methodology. Results were largely inclusive when model performances were evaluated for design methodology. It was interesting to note however that the Ada-specific model, SoftCost-Ada, which was most accurate on projects in which OOD was the specified design methodology (four out of five results were within 13%), had no effort estimates within 30% of project actuals on the two projects which utilized a structured design approach. Given the small sampling, it is difficult to attribute these results to the specified design methodology.

STRUCTURED DESIGN

**Model Accuracy.** Results were inconclusive with regard to design approach. The number of effort estimates accurate within $\pm30\%$ were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. COSTMODL | 1 out of 2 | - 1% |
| 2. SASET | 1 out of 2 | 18% |
| 3. SPQR/20 | 1 out of 2 | -20% |
| 4. SoftCost-Ada | 0 out of 2 | Not Applicable |
| 5. PRICE S | 0 out of 2 | Not Applicable |
| 6. SYSTEM-3 | 0 out of 2 | Not Applicable |

**Model Consistency.** No trends could be identified given the small sampling of data. The number of effort estimates accurate within ±30% after means were applied were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SYSTEM-3 | 2 out of 2 | 26% to 28% |
| 2. SASET | 2 out of 2 | -15% to 27% |
| 3. PRICE S | 1 out of 2 | 0% |
| 4. SPQR/20 | 1 out of 2 | -18% |
| 5. COSTMODL | 1 out of 2 | -29% |
| 6. SoftCost-Ada | 0 out of 2 | Not Applicable |

## OBJECT ORIENTED DESIGN

**Model Accuracy.** SoftCost-Ada was most accurate on projects in which OOD was the specified design methodology. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 4 out of 5 | 0% to 13% |
| 2. SASET | 3 out of 6 | -29% to 29% |
| 3. SPQR/20 | 2 out of 6 | -22% to 19% |
| 4. COSTMODL | 1 out of 4 | -25% |
| 5. PRICE S | 0 out of 6 | Not Applicable |
| 6. SYSTEM-3 | 0 out of 6 | Not Applicable |

**Model Consistency.** Results could not be correlated to the modern software development practices language considerations of the models (discussed in section 2.2.2). The number of effort estimates accurate within ±30% after means were applied were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 4 out of 5 | -13% to - 2% |
| 2. PRICE S | 4 out of 6 | -26% to 22% |
| 3. COSTMODL | 2 out of 4 | - 1% to -10% |
| 4. SYSTEM-3 | 3 out of 6 | -14% to 28% |
| 5. SASET | 2 out of 6 | - 7% to - 1% |
| 6. SPQR/20 | 2 out of 6 | -20% to 21% |

**Learning Curve: Ada Experience.** Of the eight projects targeted in the test case study, two (Projects 4 and 5) professed an average Ada experience of greater than three years. An evaluation of model performances based on Ada experience yielded results that could not be correlated to the learning curve language considerations of the models applied in the case study. Results are provided for Ada experience less than three years and greater than three years, respectively.

LESS THAN 3 YEARS OF ADA EXPERIENCE

**Model Accuracy.** No trends could be identified in light of the language considerations of the models. The number of effort estimates accurate within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SASET | 4 out of 6 | -29% to 29% |
| 2. SoftCost-Ada | 3 out of 6 | 2% to 13% |
| 3. SPQR/20 | 3 out of 6 | -22% to 19% |
| 4. COSTMODL | 2 out of 5 | -25% to 2% |
| 5. PRICE S | 0 out of 6 | Not Applicable |
| 6. SYSTEM-3 | 0 out of 6 | Not Applicable |

**Model Consistency.** Results were consistent and could not be correlated to the language considerations of the models. The number of effort estimates accurate within ±30% after means were applied were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. PRICE S | 4 out of 6 | -26% to 0% |
| 2. SASET | 4 out of 6 | -15% to 27% |
| 3. SYSTEM-3 | 4 out of 6 | -14% to 28% |
| 4. COSTMODL | 3 out of 5 | -29% to 10% |
| 5. SoftCost-Ada | 3 out of 6 | - 2% to -11% |
| 6. SPQR/20 | 3 out of 6 | -20% to 21% |

GREATER THAN 3 YEARS OF ADA EXPERIENCE

**Model Accuracy.** SoftCost-Ada was the only model accurate within the 30% range on one of the two project with personnel having greater than three years Ada experience. The numbers of effort estimates within ±30% were as follows:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 1 out of 1 | 0% |
| 2. COSTMODL | 0 out of 1 | Not Applicable |
| 3. PRICE S | 0 out of 2 | Not Applicable |
| 4. SYSTEM-3 | 0 out of 2 | Not Applicable |
| 5. SASET | 0 out of 2 | Not Applicable |
| 6. SPQR/20 | 0 out of 2 | Not Applicable |

**Model Consistency.** After means were applied to effort results, three models demonstrated performance within the ±30% range:

| MODEL | PERFORMANCE | RANGE |
|---|---|---|
| 1. SoftCost-Ada | 1 out of 1 | -13% |
| 2. PRICE S | 1 out of 2 | 22% |
| 3. SYSTEM-3 | 1 out of 2 | 28% |
| 4. COSTMODL | 0 out of 1 | Not Applicable |
| 5. SASET | 0 out of 2 | Not Applicable |
| 6. SPQR/20 | 0 out of 2 | Not Applicable |

### 3.4.3 Comparison of Nominal Results

Models were applied using nominal (average) values for input ratings while providing actual project values for model input parameters that must be estimated early in the life cycle and for which there is no associated average value. The nominal inputs reflect the level of knowledge about a new development prior to contract award. Parameters assumed to be available at contract award included the following:

1. Type of Application/Project Type
2. Estimate Scope (i.e., component within a system, stand-alone program, release, etc.)
3. Project Class
4. Programming Language
5. System Architecture *
6. Schedule *
7. Size *

(* Although size, schedule, and system architecture would not be known at contract award, an estimate would still be needed. Therefore the size, schedule, and system architecture were estimated as the actual values used in the system.) For instances when nominal ratings were not provided, parameters were examined and a nominal rating was chosen. The following assumed values pertain to specific models for which an average value was not clearly identifiable:

Ada COCOMO

**Experience with Ada Process Model.** The nominal response to this parameter was assumed to be some familiarity with practices.

**Design Thoroughness at PDR.** The nominal response to design thoroughness at PDR was assumed to be Often (60%).

**Risks Eliminated at PDR.** The nominal response was assumed to be Often (60%).

**Requirements Volatility during Development.** The nominal response to requirements volatility during development was assumed to be occasional, moderate changes.

SoftCost-Ada

**Number of Ada Projects Completed by Team.** The nominal rating associated with the number of Ada projects completed by the development team was assumed to be 1.

SASET

**Percent of Memory Utilized.** The nominal value for percent of memory utilized was assumed to be 50%.

**Development Locations.** The number of development locations was assumed to be 1 location.

As in the previous section, nominal results were examined with regard to accuracy, consistency, contract type, and application type. An analysis of results in light of Ada issues was not performed because model inputs that reflected the learning curve and modern development practices were set to their nominal default values. The results are presented in the following subsections.

### 3.4.3.1 Overall Accuracy

Nominal efforts and schedules were compared to actual efforts and schedules to ascertain which models provided the most accurate results on nominal inputs. Tables 3-11 and 3-12 show the nominal efforts and relative errors associated with the six models. The number of nominal effort estimates accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SASET | 4 out of 8 | -24% to 29% |
| 2. | SYSTEM-3 | 3 out of 8 | -17% to 28% |
| 3. | COSTMODL | 2 out of 6 | -25% to -24% |
| 4. | SoftCost-Ada | 2 out of 7 | -27% to 14% |
| 5. | PRICE S | 2 out of 8 | -14% to -8% |
| 6. | SPQR/20 | 1 out of 8 | -27% |

Nominal schedule estimates and associated relative errors are shown in Tables 3-13 and 3-14. The number of schedule estimates accurate within ±30% were as follows:

| | MODEL* | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SPQR/20 | 6 out of 8 | -23% to 28% |
| 2. | PRICE S | 4 out of 8 | -26% to 21% |
| 3. | SoftCost-Ada | 2 out of 7 | - 6% to 5% |
| 4. | SYSTEM-3 | 2 out of 8 | -23% to 5% |
| 5. | COSTMODL | 1 out of 6 | -26% |

* SASET nominal schedules were not available.

TABLE 3-11

NOMINAL EFFORT PROJECTIONS

| Number | Actual Effort | SoftCost-Ada | COSTMODL | PRICE S | SASET | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|---|
| 1 | 302 | 344 | 229 | 621 | 463 | 388 | 219 |
| 2 | 684 | 439 | 511 | 626 | 635 | 715 | 334 |
| 3 | 134 | 42 | Not Run | 36 | 102 | 70 | 219 |
| 4 | 692 | Not Run | Not Run | 2604 | 2652 | 6079 | 2562 |
| 5 | 144 | 517 | 438 | 749 | 814 | 761 | 286 |
| 6 | 190 | 109 | 94 | 529 | 394 | 131 | 100 |
| 7 | 696 | 508 | 308 | 1329 | 760 | 576 | 1069 |
| 8 | 322 | 92 | 71 | 276 | 415 | 107 | 78 |

TABLE 3-12

RELATIVE ERRORS FOR NOMINAL EFFORTS WHEN COMPARED TO ACTUAL EFFORTS

| Number | Actual Effort | SoftCost-Ada | COSTMODL | PRICE S | SASET | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|---|
| 1 | 302 | 14% | -24% | 106% | 53% | 28% | -27% |
| 2 | 684 | -36% | -25% | - 8% | - 7% | 5% | -51% |
| 3 | 134 | -69% | Not Run | -73% | -24% | -48% | 63% |
| 4 | 692 | Not Run | Not Run | 276% | 283% | 778% | 270% |
| 5 | 144 | 259% | 204% | 420% | 465% | 428% | 99% |
| 6 | 190 | -43% | -51% | 178% | 107% | -31% | -47% |
| 7 | 696 | -27% | -56% | 91% | 9% | -17% | 54% |
| 8 | 322 | -71% | -78% | -14% | 29% | -67% | -76% |

TABLE 3-13

NOMINAL SCHEDULE PROJECTIONS

| Proj. Number | Actual Schedule | SoftCost-Ada | COSTMODL | PRICE S | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|
| 1 | 27 | 28.39 | 20.0 | 36.0 | 20.9 | 33.3 |
| 2 | 38 | 35.81 | 26.3 | 46.0 | 22.1 | 42.7 |
| 3 | 35 | 10.34 | Not Run | 26.0 | 11.8 | 33.2 |
| 4 | 49 | Not Run | Not Run | 50.0 | 51.6 | 62.7 |
| 5 | 12 | 33.28 | 25.0 | 36.0 | 26.1 | 38.7 |
| 6 | 40 | 18.14 | 14.6 | 54.0 | 14.6 | 30.8 |
| 7 | 62 | 33.09 | 21.6 | 53.0 | 23.8 | 73.6 |
| 8 | 56 | 14.1 | 13.4 | 37.0 | 13.4 | 31.9 |

TABLE 3-14

RELATIVE ERRORS FOR NOMINAL SCHEDULES WHEN COMPARED TO ACTUAL SCHEDULES

| Proj. Number | Actual Schedule | SoftCost-Ada | COSTMODL | PRICE S | SYSTEM-3 | SPQR/20 |
|---|---|---|---|---|---|---|
| 1 | 27 | 5% | -26% | 33% | -23% | 23% |
| 2 | 38 | - 6% | -31% | 21% | -42% | 12% |
| 3 | 35 | -70% | Not Run | -26% | -66% | - 5% |
| 4 | 49 | Not Run | Not Run | 2% | 5% | 28% |
| 5 | 12 | 177% | 108% | 200% | 118% | 223% |
| 6 | 40 | -55% | -64% | 35% | -64% | -23% |
| 7 | 62 | -47% | -65% | -15% | -62% | 19% |
| 8 | 56 | -75% | -76% | -34% | -76% | -43% |

### 3.4.3.2 Overall Consistency

Nominal results were also checked for consistency. Table 3-15 ana 3-16 illustrate the relative errors after the calculated means were applied. The number of effort estimates accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | COSTMODL | 3 out of 6 | -23% to 30% |
| 2. | SoftCost-Ada | 3 out of 7 | 0% to 28% |
| 3. | SASET | 3 out of 8 | -24% to 7% |
| 4. | SYSTEM-3 | 3 out of 8 | -26% to 13% |
| 5. | SPQR/20 | 1 out of 8 | -14% |
| 6. | PRICE S | 1 out of 8 | -29% |

After application of the means, the number of schedule estimates accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SPQR/20 | 6 out of 8 | -28% to 20% |
| 2. | PRICE S | 5 out of 8 | -28% to 29% |
| 3. | SYSTEM-3 | 3 out of 8 | -25% to 19% |
| 4. | COSTMODL | 2 out of 6 | -27% to -23% |
| 5. | SoftCost-Ada | 2 out of 7 | 0% to 14% |

* SASET Nominal Schedules were not available for evaluation.

### 3.4.3.3 Analysis By Application Type and Contract Type

**Government Versus Commercial Contracts.** Examination of nominal performances identified the following trends:

TABLE 3-15

RELATIVE ERRORS FOR NOMINAL EFFORT AFTER APPLYING MEANS

| PROJECT NUMBER | SoftCost-Ada Mean=1.75 | COSTMODL Mean=1.74 | PRICE S Mean=0.78 | SASET Mean=0.70 | SYSTEM-3 Mean=1.08 | SPQR/20 Mean=1.18 |
|---|---|---|---|---|---|---|
| 1 | 99% | 32% | 60% | 7% | 39% | - 14% |
| 2 | 12% | 30% | - 29% | - 35% | 13% | - 42% |
| 3 | - 45% | Not Run | - 79% | - 47% | - 44% | 93% |
| 4 | Not Run | Not Run | 194% | 168% | 849% | 337% |
| 5 | 528% | 429% | 306% | 296% | 471% | 134% |
| 6 | 0% | - 14% | 117% | 45% | - 26% | - 38% |
| 7 | 28% | - 23% | 49% | - 24% | - 11% | 81% |
| 8 | 50% | - 62% | - 33% | - 10% | - 64% | - 72% |

TABLE 3-16

RELATIVE ERROR FOR NOMINAL SCHEDULE AFTER APPLYING MEANS

| PROJECT NUMBER | SoftCost-Ada Mean=1.89 | COSTMODL Mean=2.1 | PRICE S Mean=0.97 | SYSTEM-3 Mean=2.05 | SPQR/20 Mean=0.94 |
|---|---|---|---|---|---|
| 1 | 99% | 56% | 29% | 59% | 16% |
| 2 | 78% | 45% | 17% | 19% | 6% |
| 3 | - 44% | Not Run | - 28% | - 31% | - 11% |
| 4 | Not Run | Not Run | - 1% | 116% | 20% |
| 5 | 424% | 338% | 191% | 346% | 203% |
| 6 | - 14% | - 23% | 31% | - 25% | - 28% |
| 7 | 0% | - 27% | - 17% | - 21% | 12% |
| 8 | - 52% | - 50% | - 36% | - 51% | - 46% |

**Model Accuracy.** SASET provided the most accurate results for government contracts where three of four estimates were within 28% of the actual effort. The number of effort estimates accurate within ±30% were as follows:

GOVERNMENT CONTRACTS

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SASET | 3 out of 4 | - 7% to 29% |
| 2. | PRICE S | 2 out of 4 | -14% to -8% |
| 3. | SYSTEM-3 | 2 out of 4 | -17% to 5% |
| 4. | COSTMODL | 1 out of 4 | -25% |
| 5. | SoftCost-Ada | 1 out of 4 | -27% |
| 6. | SPQR/20 | 0 out of 4 | Not Applicable |

**Model Consistency.** SofCost-Ada, Ada COCOMO, and SYSTEM-3 all provided three of four estimates to within 30% of the actual effort expended. The number of effort estimates accurate within ±30% after calculated means were applied were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SoftCost-Ada | 3 out of 4 | 0% to 28% |
| 2. | SYSTEM-3 | 3 out of 4 | -26% to 13% |
| 3. | COSTMODL | 3 out of 4 | -23% to 30% |
| 4. | SASET | 2 out of 4 | -24% to -10% |
| 5. | PRICE S | 1 out of 4 | -29% |
| 6. | SPQR/20 | 0 out of 4 | Not Applicable |

COMMERCIAL CONTRACTS

**Model Accuracy.** All models performed equally on commercial contracts so no trends could be identified. The number of effort estimates accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | COSTMODL | 1 out of 2 | -24% |
| 2. | SoftCost-Ada | 1 out of 3 | 14% |
| 3. | SASET | 1 out of 4 | -24% |
| 4. | SPQR/20 | 1 out of 4 | -27% |
| 5. | SYSTEM-3 | 1 out of 4 | 28% |
| 6. | PRICE S | 0 out of 4 | Not Applicable |

**Model Consistency.** No trends could be identified for commercial contracts. The number of effort estimates accurate within ±30% after means were applied were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SASET | 1 out of 4 | 7% |
| 2. | SPQR/20 | 1 out of 4 | -14% |
| 3. | SoftCost-Ada | 0 out of 3 | Not Applicable |
| 4. | PRICE S | 0 out of 4 | Not Applicable |
| 5. | COSTMODL | 0 out of 2 | Not Applicable |
| 6. | SYSTEM-3 | 0 out of 4 | Not Applicable |

**Application Type.** Nominal Performances in light of application type were as follows:

COMMAND AND CONTROL

**Model Accuracy.** SASET provided the most accurate results with three of its four estimates within 24% of the actual effort. The number of effort estimates accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SASET | 3 out of 4 | - 7% to 29% |
| 2. | SYSTEM-3 | 3 out of 4 | -17% to 28% |
| 3. | COSTMODL | 2 out of 4 | -25% to -24% |
| 4. | PRICE S | 2 out of 4 | -14% to - 8% |
| 5. | SoftCost-Ada | 2 out of 4 | -27% to 14% |
| 6. | SPQR/20 | 1 out of 4 | -27% |

**Model Consistency.** SASET provided the most accurate results with three of four estimates accurate to within 24% of the actual effort. The number of effort estimates accurate within ±30% after means were applied were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SASET | 3 out of 4 | -24% to 7% |
| 2. | SoftCost-Ada | 2 out of 4 | 12% to 28% |
| 3. | SYSTEM-3 | 2 out of 4 | -11% to 13% |
| 4. | COSTMODL | 2 out of 4 | -23% to 30% |
| 5. | SPQR/20 | 1 out of 4 | -14% |
| 6. | PRICE S | 1 out of 4 | -29% |

TOOLS/ENVIRONMENT

**Model Accuracy.** All models performed equally on these types of systems in the nominal runs. The number of effort estimates accurate within ±30% were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | SASET | 1 out of 3 | -24% |
| 2. | SoftCost-Ada | 0 out of 2 | Not Applicable |
| 3. | COSTMODL | 0 out of 1 | Not Applicable |
| 4. | PRICE S | 0 out of 3 | Not Applicable |
| 5. | SYSTEM-3 | 0 out of 3 | Not Applicable |
| 6. | SPQR/20 | 0 out of 3 | Not Applicable |

**Model Consistency.** All models also performed equally. The number of effort estimates accurate within ±30% after calculated means were applied were as follows:

| | MODEL | PERFORMANCE | RANGE |
|---|---|---|---|
| 1. | COSTMODL | 0 out of 1 | Not Applicable |
| 2. | SoftCost-Ada | 0 out of 2 | Not Applicable |
| 3. | PRICE-S | 0 out of 3 | Not Applicable |
| 4. | SASET | 0 out of 3 | Not Applicable |
| 5. | SYSTEM-3 | 0 out of 3 | Not Applicable |
| 6. | SPQR/20 | 0 out of 3 | Not Applicable |

This page is intentionally left blank.

## 4.0 DATA ANALYSIS

### 4.1 BACKGROUND

Once data was validated and normalized, six projects targeted in the study were statistically analyzed to determine how they compared to the normal productivity ranges exhibited by other Ada projects within an application domain. Then, cost drivers inherent in the data itself were investigated via statistical means (i.e., hypothesis testing, goodness of fit, etc.) to validate cost drivers previously identified and to uncover any new cost drivers.

A report prepared for IITRI, entitled "Ada Data Analysis and Normalization" (RCI-TR-065), dated 9 January 1989, contains results of the data analysis task that was performed by Reifer Consultants, Inc. (RCI). The report identifies primary cost drivers for 10 application areas categorized as follows:

1. Automation
2. Command and Control
3. Data Processing
4. Environment
5. Military (Embedded)
6. Scientific
7. Simulation
8. Telecommunications
9. Test
10. Other

Statistical sensitivity analysis was conducted on the domain of primary cost drivers to see if the new Ada project data fit existing relationships developed previously by RCI. Data that did not fit would have an enlarged variance when compared to the norms in each of the domains. Hypothesis testing was conducted to identify new cost drivers. The result of the analyses was an updated list of cost drivers in each of the 10 application areas.

## 4.2 RESULTS

### Domain Analysis

The new data provided by the six projects fell within the normal ranges for those projects already within the RCI Ada database for the application domain noted in Table 4-1.

**TABLE 4-1**

**RESULT OF DOMAIN ANALYSIS [REIF89]**

| Project | Type of System (Domain) | Productivity SLOC/pm | Productivity Range (mean) | Variance (SLOC/pm) |
|---|---|---|---|---|
| 1 | Automation | 173 | 112-345 (210) | ± 33 |
| 2 | Command & Control | 190 | 26-289 (180) | ± 30 |
| 3 | Environmental/Tool | 113 | 81-587 (240) | ± 26 |
| 4 | Environmental/Tool | 400 | 81-587 (240) | ± 26 |
| 5 | Environmental/Tool | 450 | 81-587 (240) | ± 26 |
| 6 | Military (Airborne) | 127 | 18-228 (143) | ± 18 |

### Sensitivity Analysis

Cost driver information provided from the questionnaires plus 10 other projects acquired from separate sources were used to perform a sensitivity analysis. The new data suggested several alterations to existing parameter calibrations and relationships derived from previous data. A summary of primary cost drivers in RCI's Ada database is presented in Table 4-2 [REIF89].

TABLE 4-2

PRIMARY COST DRIVERS IN RCI'S ADA DATABASE [REIF89]

| Type of Software | Number of Projects | Size Range (ASLOC) | Primary Cost Drivers |
|---|---|---|---|
| Automation | 8 | 5-120 | - Requirements Volatility<br>- Reuse<br>- Experience |
| Command & Control | 12 | 25-1800 | - Requirements Volatility<br>- Reuse<br>- Degree of Standardization<br>- Experience<br>- Architecture |
| Data Processing | 10 | 25-450 | - Requirements Volatility<br>- Reuse<br>- Experience<br>- Architecture |
| Environment/Tools | 22 | 5-1000 | - Requirements Volatility<br>- Reuse<br>- Experience |
| Military (Embedded) | 12 | 2-250 | - Requirements Volatility<br>- Complexity<br>- Reuse<br>- Degree of Standardization<br>- Degree of Real-Time<br>- Experience |
| Scientific | 4 | 28-300 | - Requirements Volatility<br>- Complexity<br>- Reuse<br>- Experience |
| Simulation | 3 | 80-175 | - Requirements Volatility<br>- Reuse<br>- Experience |

TABLE 4-2

PRIMARY COST DRIVERS IN RCI'S ADA DATABASE (Continued)

| Type of Software | Number of Projects | Size Range | Primary Cost Drivers |
|---|---|---|---|
| Telecommunications | 12 | 5-400 | - Requirements Volatility<br>- Complexity<br>- Reuse<br>- Experience<br>- Architecture |
| Test | 3 | 22-125 | - Requirements Volatility<br>- Reuse<br>- Experience |
| Other | 7 | 5-180 | - Requirements Volatility<br>- Reuse<br>- Experience |

## 5.0 CONCLUSIONS

The following six cost estimation models were applied to a database of eight completed Ada projects:

| Ada-Specific Models | | Non-Ada Specific Models | |
|---|---|---|---|
| 1. | Ada COCOMO (IOC) | 1. | PRICE-S (188230) |
| 2. | SoftCost-Ada (1.3) | 2. | SASET (1.5) |
| | | 3. | SPQR/20 (1.2) |
| | | 4. | SYSTEM-3 (1.03) |

The version of each model reviewed is indicated in parenthesis beside the model name. Project questionnaires were completed by the developers. This information was used to derive inputs for each model. Emphasis was placed on providing a consistent set of inputs across all models. In addition, models were applied using nominal (average) values for input ratings while providing actual project values for size, application type, programming language, and other model inputs that must be estimated early in the life cycle and for which there is no associated average value. The nominal inputs reflect the level of knowledge about a new project prior to contract award. Resultant analyses were based on a comparison of each model's schedule and effort projection and nominal run results to the actual effort expended by the software developer.

Results were evaluated for accuracy and consistency for each of the following six categories:

1. Overall effort
2. Overall schedule
3. Government contracts
4. Commercial contracts
5. Command and control applications
6. Tools and environment applications.

Tables 5-1 and 5-2 respectively summarize the test case study results for derived and nominal inputs. For each evaluation criteria, the two models that had the highest performance ratings are listed. Model results were also evaluated based on the project's design approach and personnel experience with Ada. Model performances could not be correlated to the language considerations of the models which are described in Section 2.

## 5.1 RECOMMENDATIONS

The test case study results demonstrate the benefits of cost models that assist the estimator in predicting resource requirements for a new development. The results do not validate the need for Ada-specific models. Although SoftCost-Ada was most accurate overall, non-Ada-specific models were comparable in terms of accuracy and consistency.

The results do recommend that users consider the following to determine which models should be applied to estimate Ada software costs:

- **Assess how much information is available about the project and the developing organization.** Application of models to both regular and nominal runs in the study showed that model performances varied with differing amount of project information. Some performed better with minimum information while others performed better with detailed information.

- **Consider the customer.** Model performances were evaluated based on the type of contract. Some models were more effective when applied to government contracts while others were more accurate for estimating commercial contracts.

- **Consider the type of application.** Projects targeted in the test case study consisted of three different types of applications: command and control (4 projects), tools/environment (3 projects), and avionics (1 project). An analysis of results based on application type revealed that models that were most accurate on command and control applications were not as accurate for tools and environment applications.

TABLE 5-1

SUMMARY TEST CASE STUDY RESULTS:
BEST TWO PERFORMANCES IN EACH CATEGORY

| Evaluation Criteria | Model | Performance (Within 30%) | Range |
|---|---|---|---|
| Overall Accuracy of Effort | SoftCost-Ada<br>SASET | 4 out of 7<br>4 out of 8 | 0% to 13%<br>-29% to -29% |
| Overall Accuracy of Schedule | SYSTEM-3<br>PRICE S | 4 out of 8<br>3 out of 8 | -27% to - 7%<br>3% to 18% |
| Overall Consistency of Effort | SYSTEM-3<br>PRICE S | 5 out of 8<br>5 out of 8 | -14% to 28%<br>-26% to 22% |
| Overall Consistency of Schedule | SYSTEM-3<br>PRICE S | 5 out of 8<br>5 out of 8 | 0% to 28%<br>-29% to 28% |
| Model Accuracy on Government Contracts | SASET<br>COSTMODL | 3 out of 4<br>2 out of 4 | - 7% to 29%<br>-25% to - 1% |
| Model Consistency on Government Contracts | SYSTEM-3<br>PRICE S | 4 out of 4<br>3 out of 4 | -14% to 28%<br>-26% to 0% |
| Model Accuracy on Commercial Contracts | SoftCost-Ada<br>SPQR/20 | 3 out of 3<br>1 out of 4 | 0% to 6%<br>-22% |
| Model Consistency on Commercial Contracts | SoftCost-Ada<br>PRICE-S | 3 out of 3<br>2 out of 4 | -13% to - 8%<br>- 1% to 22% |
| Model Accuracy on Command & Control Applications | SASET<br>SPQR/20 | 3 out of 4<br>3 out of 4 | - 7% to 29%<br>-22% to 19% |
| Model Consistency on Command & Control Applications | PRICE S<br>SASET | 4 out of 4<br>3 out of 4 | -26% to 0%<br>-15% to 1% |
| Model Accuracy on Tools/Environment Applications | SoftCost-Ada<br>SASET | 2 out of 2<br>1 out of 3 | 0% to 2%<br>-29% |
| Model Consistency on Tools/Environment Applications | SoftCost-Ada<br>SYSTEM-3 | 2 out of 2<br>1 out of 3 | -13% to -11%<br>28% |

TABLE 5-2

SUMMARY TEST CASE STUDY RESULTS FOR NOMINAL RUNS:
BEST TWO PERFORMANCES IN EACH CATEGORY

| Evaluation Criteria | Model | Performance (Within 30%) | Range |
|---|---|---|---|
| Overall Accuracy of Effort | SASET<br>SYSTEM-3 | 4 out of 8<br>3 out of 8 | -24% to 29%<br>-17% to 28% |
| Overall Accuracy of Schedule | SPQR/20<br>PRICE S | 6 out of 8<br>4 out of 8 | -23% to 28%<br>-26% to 21% |
| Overall Consistency of Effort | COSTMODL<br>SoftCost-Ada | 3 out of 6<br>3 out of 7 | -23% to 30%<br>0% to 28% |
| Overall Consistency of Schedule | SPQR/20<br>PRICE S | 6 out of 8<br>5 out of 8 | -28% to 20%<br>-28% to 29% |
| Model Accuracy on Government Contracts | SASET<br>PRICE S | 3 out of 4<br>2 out of 4 | - 7% to 29%<br>-14% to - 8% |
| Model Consistency on Government Contracts | SoftCost-Ada<br>SYSTEM-3 | 3 out of 4<br>3 out of 4 | 0% to 28%<br>-26% to 13% |
| Model Accuracy on Commercial Contracts | COSTMODL<br>SoftCost-Ada | 1 out of 2<br>1 out of 3 | -24%<br>14% |
| Model Consistency on Commercial Contracts | SASET<br>SPQR/20 | 1 out of 4<br>1 out of 4 | 7%<br>-14% |
| Model Accuracy on Command & Control Applications | SASET<br>SYSTEM-3 | 3 out of 4<br>3 out of 4 | - 7% to 29%<br>-17% to 28% |
| Model Consistency on Command & Control Applications | SASET<br>SoftCost-Ada | 3 out of 4<br>2 out of 4 | -24% to 7%<br>-12% to 28% |
| Model Accuracy on Tools/Environment Applications | SASET | 1 out of 3 | -24% |
| Model Consistency on Tools/Environment Applications | | 0 | |

## 5.2 LESSONS LEARNED

While the study was performed on six cost estimation models, there are many other models that are currently being used to estimate Ada software costs. Additionally, as Ada usage and technology increase, model developers will be constantly refining their models to provide more accurate results in this field. Two models reviewed in this study, SoftCost-Ada and SYSTEM-3 (now SYSTEM-4), have since released new versions of their models. Ada COCOMO is an initial operational capability that will be subject to further refinement. Because of these factors, it is important to identify lessons learned during the test case study to facilitate further research in this area.

**Lesson No. 1:** Have a specific objective in mind when you approach a developer for data; Use a simple, concise data collection form that focuses only on the collection objective.

Many requests were made to individuals to provide data on a voluntary basis for the test case study. A specific objective provided an incentive to the developer to provide data, especially if the developer knew what the data was being used for and that results of the data analysis would be valuable to his organization. The test case study results were of great interest to all of the developers who provided data for the research.

A single objective also focuses collection activities for the developing organization. Developers are more likely to provide data if it won't require too much effort. Requesting a broad range of information from a single organization (i.e., cost data, quality data, productivity data) will likely result in losing a potential data resource.

Ada project data was provided for each project in the form of a completed project questionnaire. Initially, Software Project Data

Collection Forms and Instructions compiled for the Comptroller's Office (ESD/ACCR) were sent to developers. These forms were tailored specifically to the Ada language and included many of the data items required to run some of the major software cost and schedule estimating models. Three projects targeted in the test case study, attained from ESD, were already in this format. However, the feedback from project developers regarding the volume of information requested caused IITRI to develop new, shorter forms devised from an examination of model input parameters.

It is also necessary to cite the minimum data that is needed before the developer uses valuable time and effort to complete the forms. For example, at minimum, the study required project effort in person months and project size in SLOC. The "We'll take anything you've got" attitude results in too little data that is too general to fulfill any useful purpose.

No matter how simple or well defined a form is, raw data provided by the developer needs to be validated. Through validation procedures, inconsistencies between responses can be identified. It is also a good idea to discuss the completed questionnaire with the person who completed the form and "spot-check" certain questions to ensure that the developer had the correct context in mind.

**Lesson No. 2:** Expect a one in five response from developers who initially agree to provide project data.

Inquiries into the projects listed in the AJPO Ada Usage Database resulted in 19 respondents who agreed to provide data for the test case study. Of these newly identified projects, four were eventually able to provide requested data in a timely manner. Reasons for not providing data included unavailability of information requested, the amount of effort that completing the form would entail, and nonapproval by project offices to release the data.

From this, we have learned the importance of targeting a large number of organizations for a specific collection objective. IITRI's status as a not-for-profit research institute and non-competitor also enhanced our position to acquire data.

**Lesson No. 3:** Provide an incentive for the developer to provide data.

Projects targeted in the test case study were obtained with the incentive that the results of the test case study would be automatically provided at no cost to the developer. In specific cases other incentives were utilized. For example, one developer did an excellent job of completing the project questionnaire but omitted the number of person months required to complete the effort. The project data that was provided by the developer could not fulfill its intended purpose unless the actual project effort was also included in the data. Upon contacting the developer, we were told that the information was proprietary. Because the project was funded by the government, the information was eventually obtained through government intervention.

In all cases, the ability to gain the cooperation of software developers required diligence on following-up initial requests and on an agreeable attitude.

**Lesson No. 4:** Identify what needs to be collected at the beginning of a new project and have the developer provide the information at the project's completion.

It is interesting to note that six of the eight projects evaluated for the study indicated extremely tight schedule constraints. In view of these accelerated schedules, data acquisition cannot be considered to be a constant nuisance to the developer. However, the developer needs to be aware of the information that needs to be collected so that it can be tracked as the project progresses. Information may be too difficult to retrieve at the point of the project's completion. For example, the Air Force wanted IITRI to evaluate the distribution of project effort by life-cycle phase. However, none of the developers who provided data for the costing study tracked effort expended by phase. It would have been beneficial to the costing study to identify data items at the beginning of each project, such as in the Request for the Proposal (RFP).

**Lesson No. 5:** The best way to gather consistent information to provide collection tools.

The Ada costing study emphasized the different conventions that various developers use to measure software attributes. Developers who provided data used one of three methods to count SLOC:

1. Physical lines including comments and blank lines
2. Physical lines excluding comments and blank lines
3. Terminal semicolons including those used in data declarations and formal parameter lists.

In order to normalize the line counts so that comparable information would be evaluated in the data analysis, conversion factors were used to standardize the line counts. For example, deciding that an Ada source line of code should be defined using terminal semicolons, physical line counts (excluding comments and blank lines) were reduced by 20% for a comparable measure of size. Obviously, the need to derive conversion factors would not have been required if an automated line-counting tool had been provided to each developer.

**Lesson No. 6:** For validation purposes, it is best to obtain project data directly from the developer rather than from the government contact.

It is best to communicate directly with the developer, for it is the developer who can answer questions about the nature of the development. It is also important that personnel at the proper level be designated to respond. To illustrate this point, IITRI had occasion to ask three people on the same project to rate the capability of the programming staff: the section manager, the project manager, and one of the programming staff. Each respondent had a different perspective on the development and each replied with a different evaluation. It is important to be aware of these unique perspectives and when feasible, use consistency when targeting project personnel for data collection.

**Lesson No. 7:** Designate a single collector and focal point for all interface with the developer.

One method that will result in more accurate data is to designate a single focal point for interface with the software developer. This approach enables the acquisitioner to become familiar with the project and be able to spot inconsistencies in the project data. Because the acquisitioner is more familiar with the development, there is a tendency to ask less questions required to familiarize oneself to a development. A single interface also allows a good rapport to develop between developer and data acquisitioner. After assessing these factors, models which meet the given criteria for the chosen project should be applied to generate cost estimates.

## 5.3 TOPICS FOR FURTHER RESEARCH

This report represents a small window in time with respect to the state of the art in software cost estimating. Case studies are beneficial because they provide definitive data that reflect on current practices. As long as Ada development methodologies continue to evolve, new case studies are warranted. We recommend that future case studies consider the following:

- **Assessment of Ada issues:** An evaluation of project data in light of Ada issues described in Section 2.2 showed that six of the eight issues could not be evaluated due to a lack of data. For example, the distribution of effort by life-cycle phase could not be evaluated because none of the developers who provided data for this study tracked effort expended by phase. Schedule and effort distribution for Ada versus non-Ada projects is a significant Ada issue. Future case studies should consider these issues and target the data that is needed to evaluate them.

- **Inclusion of other models:** Models that were included in this study were selected based upon their availability to either the AFCSTC, USACEAC, or IITRI. Model vendors were not solicited. There are however a number of other good models, for example, SLIM and SEER, that researchers should consider evaluating in future case studies.

- **Data acquisition that targets specific application types:** Findings indicate some interesting trends based on the type of contract - commercial or government - and the project's application type. Future studies should consider these trends and target projects based on these evaluation criteria. For example, it would be beneficial to evaluate model performance for business applications.

- **Maintenance cost evaluation:** It is generally perceived that Ada usage will result in long-term benefits of reduced costs required to maintain code that is less error prone. Cost models usually provide life-cycle maintenance cost for a specified term. It would be useful to compare predicted costs to actual Ada maintenance experience.

## APPENDIX A

## MODEL VENDORS/POINTS OF CONTACT (POC)

Each of the models included in this study are undergoing continual revision as developers receive feedback from their users. For additional information about a model or package, the designated vendor/point of contact listed in Table A-1 should be contacted.

TABLE A-1

MODEL VENDORS/POINTS OF CONTACT (POC)

| MODEL | VENDOR/POC |
|---|---|
| Ada COCOMO | Dr. Barry Boehm<br>TRW Defense Systems Group<br>One Space Park<br>Redondo Beach, CA 90278<br>(213) 812-0786 |
| PRICE S | Dr. Robert E. Park<br>PRICE Systems<br>General Electric Company<br>300 Route 38, Bldg. 146<br>Moorestown, NJ 08057<br>1-800-GE-PRICE |
| SASET | Mr. Steve Gross<br>Naval Center for Cost Analysis<br>Department of the Navy<br>Washington, DC 20350-1100<br>(202) 694-0173 |
| SoftCost-Ada | Mr. Donald Reifer<br>Reifer Consultants, Inc.<br>25550 Hawthorne Blvd, Suite 208<br>Torrance, CA 90505<br>(213) 373-8728 |

TABLE A-1 (Continued)

COST MODEL POINTS OF CONTACT

| | |
|---|---|
| SPQR/20 | Mr. Wayne Hadlock<br>Software Productivity Research, Inc.<br>P.O. Box 1033<br>1972 Massachusetts Avenue<br>Cambridge, MA 02140<br>(617) 495-0120 |
| SYSTEM-3 | Mr. Wayne Stanley<br>Computer Economics, Inc.<br>Suite 109<br>4560 Admiralty Way<br>Marina del Rey, CA 90292-5424<br>(213) 827-7300 |
| DoD: | Lt. Paul Marsey<br>Wright-Patterson AFB<br>(513) 255-6347 |

TABLE A-2

ADA COCOMO IMPLEMENTATIONS POINTS OF CONTACT (POC)

| PACKAGE | POC |
|---|---|
| BMO* | Lt. Darrish<br>Headquarters BMO-ACS<br>Norton AFB, CA 92409-6468<br>(714) 382-4713 Autovon: 876-5836 |
| COSTAR | Mr. Dan Ligett<br>Softstar Systems<br>28 Ponemah Road<br>Amherst, NH 03031<br>(603) 672-0987 |
| COSTMODL | Mr. Bernie Roush<br>NASA Johnson Space Center<br>Mail Code FM 7<br>Houston, TX 77058<br>(713) 483-9092 |
| GECOMO | Ms. Susan Boers<br>GEC Software<br>1850 Centennial Park Drive, Suite 300<br>Reston, VA 22091<br>(703) 648-1551 |
|  | Mr. Peter Sizer<br>132-135 Long Acre<br>London WC2E England<br>44-1-240-7171 |

* Currently does not include Incremental Development.
Restricted use to Government only.

This page is intentionally left blank.

# APPENDIX B

## MODEL INPUT PARAMETERS: DETAILED DEFINITIONS

This appendix contains a detailed description of model input parameters for automated models discussed in Section 2.0. The input parameter descriptions are listed by model name.

(1) Ada COCOMO

(2) PRICE S

(3) SASET

(4) SoftCost-Ada

(5) SPQR/20

(6) SYSTEM-3

Input variables for each model are categorized by input type as follows:

- Product attributes describe the characteristics of the software to be developed.
- Sizing factors determine the quantity of software to be developed.
- Process attributes describe the development methodology: requirements definition, use of software tools, modern programming practices, schedule, supporting documentation, etc.
- Computer attributes refer to the virtual machine underlying the system to be developed.
- Personnel attributes address the experience and capabilities of the software development team assigned to the project.
- Environmental factors describe other external circumstances that affected the development such as security requirements, the makeup of the organizations involved, development locations, and office facilities.

The description for each parameters contains the following information:

- Name of input variable
- Type of input - Input value formats are categorized as follows:

|  |  |
|---|---|
| R = | Rating |
| % = | Percentage or proportion |
| # = | Count |
| C = | Category |
| S = | Character string |
| Y/N = | Yes or No |
| E = | Empirically derived |

- Default value
- Required or optional input
- Input variable definition - Descriptions provided here are summary explanations. More detailed definitions and additional clarification of inputs are provided in the training for use of the model or in the model's reference/user's manual.

The format for each variable description is as follows:

Name [type, default, required or optional] - input variable definition.

## Ada COCOMO

Product Attributes

1. **Required software reliability** [R, Default = nominal, Required] - degree of reliability required for a software subsystem measured in terms of the effect of a software failure.

2. **Software product complexity** [R, Default = nominal, Required] - level of complexity of the module to be developed as a function of the type of operations to be performed by the module: control, computation, device-dependent, or data management.

3. **Required reusability*** [C, Default = Ada components not for reuse elsewhere, Required] - the extent that the developer must design, document, and test Ada components for reuse in another application.

Sizing Factors

1. **Delivered source instructions** [#, No default, Required]- count of all program instructions created by project personnel based on the number of carriage returned in package specifications plus the number of semicolons in package bodies.

2. **Adapted or reused instructions** [#, No default, Required]- count of **delivered source instructions** adapted from existing software to form the new product.

   Adapted Code Modification Factors:

3. **Design modified** [%, No default, Required] - percentage of the adapted software's design which is modified in order to adapt it to the new project's objectives and environment.

4. **Code modified** [%, No default, Required] - percentage of the adapted software's code which is modified in order to adapt it to the new project's objectives and environment.

5. **Integration required for modified software** [%, No default, Required] - amount of effort required to integrate and test the adapted software into an overall product as compared to the normal amount of integration and test effort for software of comparable size.

* New cost driver for Ada COCOMO

6. **Data base size** [R, Default = nominal, Required] - ratio of amount of data to be assembled and stored in main storage by the time of software acceptance to the total program size in delivered source instructions.

Process Attributes

1. **Experience with Ada Process Model*** [R, No default, Required] -rating of the amount of familiarity and use of the TRW Ada Process Model.

2. **Design thoroughness at PDR: Unit package specs compiled, bodies outlined*** [R, No default, Required] - degree to which compilable package specifications (and body outlines) were produced for all top-level and critical lower-level Ada packages by PDR.

3. **Risks eliminated by PDR*** [R, No default, Required] - degree to which all major risk items are identified and eliminated by PDR.

4. **Requirements volatility during development*** [R, No default, Required] - frequency and criticality of changes in requirements during product development.

5. **Use of modern programming practices** [R, Default = nominal, Required] - degree to which modern programming practices are used in developing the software.

6. **Use of software tools** [R, Default = nominal, Required]- degree to which software tools are used in developing the software subsystem.

7. **Required development schedule** [R, Default = nominal, Required] - level of schedule constraint imposed on the project team developing a software subsystem expressed in terms of the percentage of schedule stretchout or acceleration with respect to a nominal schedule with nominal effort.

Computer Attributes

1. **Execution time constraint** [R, Default = nominal, Required]- degree of execution time constraint imposed upon a subsystem expressed in terms of the percentage of available execution time expected to be used by the subsystem and any other subsystem consuming the execution resources.

* New cost driver for Ada COCOMO

2. **Main storage constraint** [R, Default = nominal, Required]- degree of main storage constraint imposed on a software subsystem expressed in terms of the percentage of main storage expected to be used by the subsystem and any other subsystems consuming the main storage resources.

3. **Virtual machine volatility: Host*** [R, Default = nominal, Required] - relative frequency of major changes and minor changes to the host virtual machine.

4. **Virtual machine volatility: Target*** [R, Default = nominal, Required] - relative frequency of major changes and minor changes to the target virtual machine.

5. **Computer turnaround time** [R, Default = nominal, Required]- average response time in hours from the time the developer submits a job to be run until the results are back in the developer's hands.

Personnel Attributes

1. **Analyst capability** [R, Default = nominal, Required] - level of capability of the analysts expressed in terms of percentiles with respect to the overall population of analysts.

2. **Programmer capability** [R, Default = nominal, Required]- level of capability of the analysts expressed in terms of percentiles with respect to the overall population of analysts.

3. **Applications experience** [R, Default = nominal, Required]- the project team's equivalent level of experience with the type of application to be developed.

4. **Virtual machine experience** [R, Default = nominal, Required]- the project team's equivalent level of experience with the underlying virtual machine for an application.

5. **Programming language experience** [R, Default = nominal, Required] - the project team's equivalent level of experience with the programming language to be used.

Environmental Factors

1. **Classified security application*** [R, Default = nominal, Required] -Unclassified (nominal rating) or classified (high) project security classification.

* New cost driver for Ada COCOMO

## PRICE S

### Product Attributes

1. **Platform** [R, No default, Required] - measure of specifications that must be met regarding the portability, reliability, structuring, testing, and documentation required for acceptable contract performance.

2. **Language** [S, No default, Required] - source language to be used for the software development effort.

3. **HSI complexity** [R, Default = 1.0, Required] - relative effect of complicating factors on the software development task caused by hardware/software interactions.

4. **Source code mix** [%, No default, Required] - percentage of deliverable **source lines of code** that performs each of the following categories of operations:

   - **Data storage and retrieval**
   - **On-line communications**
   - **Real-time command and control**
   - **Interactive operations**
   - **Mathematical applications**
   - **String manipulations**
   - **Operating systems**, and/or

   - **Other.**

5. **User defined application** [R, No default, Required if a percentage of **source lines of code** is allocated to the **Other** category of operation] - describes the instruction mix of software corresponding to **Other** (i.e. diagnostics, array processing, quad-redundant operating systems, etc.).

### Sizing Factors

1. **Source lines of code** [#, No default, Required] - total number of source lines of code to be developed excluding comments.

2. **Non-executable code** [%, No default, Required] - fraction of source lines of code describing type declarations, data statements, data initializations, and instantiations.

3. **New design** (corresponding to source code mix) [%, No default, Required] - amount of new design required for the amount of software for each category of operation designated under **source code mix.**

4. **New code** (corresponding to source code mix) [%, No default, Required] - amount of new coding required for the amount of software for each category of operation designated under **source code mix.**

Process Attributes

1. **Internal integration** [R, No default, Required] - level of difficulty of integrating and testing software components to the CSCI level.

2. **External integration** [R, No default, Required only if system level integration is to be estimated] - level of difficulty of integrating and testing the CSCIs to the system level.

3. **Schedule** [S, No default, Either SDR or SSR dates required, other dates are optional] - Milestone dates relative to the single CSCI being estimated:

   - System Concept effort starts
   - System Requirements Review
   - System Design Review (SDR)
   - Software Specification Review (SSR)
   - Preliminary Design Review
   - Critical Design Review
   - Test Readiness Review
   - Functional Configuration Audit
   - Physical Configuration Audit
   - Formal Qualification Review
   - Operational Test and Evaluation.

   PRICE S generates a schedule based on user-supplied data in addition to an nominal or typical schedule based on the SDR or SSR date that is input.

Computer Attributes

1. **Utilization** [%, No default, Required] - fraction of available hardware cycle time or total memory capacity used.

Environment Factors

1. **Management complexity** [R, No default, Required] - relative effect of complicating factors on the software task such as development at more than one location and/or a multinational project.

2. **Productivity factor** [E, No default, Required] - net effect of organizational characteristics including such factors as skill levels, experience, productivity, and efficiency.

3. **Complexity** [R, Default = 1.0, Required] - relative effect of complicating factors on the software development tasks including such factors as product familiarity, personnel skills, software tools, new language, and changing requirements.

## SASET

### Product Attributes

1. **Class of software** [C, No default, Required] - operating environment of the software to be developed: manned flight, unmanned flight, avionics, shipboard/submarine, ground, commercial.

2. **Primary Software Language** [C, No default, Required]- principle software language that is used to develop the software system.

3. **Man interaction** [R, Default = Average, Required] - level of man interaction inherent in the system.

4. **Timing and Criticality** [R, Default = Average, Required]- performance requirements related to response times.

5. **Software testability** [R, Default = Average, Required] - level of difficulty associated with extent and ease of software testing.

6. **Software interfaces** [R, Default = Average, Required] - number of external software interfaces to other programs, systems, and/or peripheral communications equipment required by the software system.

7. **Embedded development system** [R, No default, Required]- extent to which hardware will be developed concurrently with the software.

8. **Programming language complexity** [R, No default, Required]- complexity of the principle software programming language rated according to the difficulty of language constructs, syntax, and required training.

### Sizing Factors

Note: SASET will estimate size according to the user defined functionality of a software system based upon the associated database. The user may optionally bypass the sizing by software functionality and directly input sizing information.

Items 1 through 5 refer to the "Software Functionality" inputs.

1. **Software Function** [C, No default, Required for functional sizing] - lowest level of decomposition for the component of the software system to be developed. The user enters quantitative parameters (items 2 through 5) for each software function defined.

2. **Size value** [R, No default, Required for functional sizing]- perceived complexity and size relative to the base size estimate provided for the identified software function.

3. **Generated/operational function** [C, No default, Required for functional sizing] - indicates whether the software function is provided by some element of the operational environment or if the function needs to be designed and developed from scratch by the software developer.

4. **Code condition** [%, No default, Required for functional sizing] - the portion of software function that is a new function, modified, or a rehosted function requiring a smaller portion of the life cycle for implementation than a modified function.

5. **Language** [%, No default, Required for functional sizing]- percent of the software function which is to be implemented in high-order language and the percent to be implemented in assembly language.

6. **CPU Host** [#, No default, Required for functional sizing] -CPU number of the total number of system CPUs on which the software function is to be hosted.

Note: Items 7 through 13 refer to direct sizing inputs.

7. **Software type** [C, No default, Required for direct sizing]- one of four software types (system, application, support, and security) for which the condition of the code (new, modified, rehosted items 8 through 13) must be identified.

8. **New HOL code** [#, No default, Required for direct sizing]- amount of HOL software that is to be developed from scratch.

9. **Modified HOL code** [#, No default, Required for direct sizing] - amount of HOL software which has some development already complete and requires retesting, some redesign and recoding effort in order to be utilized.

10. **Rehosted HOL code** [#, No default, Required for direct sizing] - amount of completed and tested HOL software code which is to be transferred from one computer system to another.

11. **New Assembly code** [#, No default, Required for direct sizing] - amount of assembly code that is to be developed from scratch.

12. **Modified Assembly code** [#, No default, Required for direct sizing] - amount of assembly code which has some development already complete and which can be utilized in the software program under consideration.

13. **Rehosted Assembly code** [#, No default, Required for direct sizing] - amount of assembly software which has some development already complete and requires retesting, some redesign and recoding effort in order to be utilized.

Process Attributes

1. **Schedule** [S, No default, Optional] - Start date, finish date, and date of Critical Design Review relative to the single CSCI being estimated. SASET generates a schedule based on user-supplied data in addition to an "optimal" schedule based on the start date that is input.

2. **System requirements** [R, Default = Average, Required] - extent to which system requirements are well defined and understood.

3. **Software requirements** [R, Default = Average, Required]- extent to which software requirements are well defined and understood.

4. **Software documentation** [R, Default = Average, Required]- level and volume of software documentation required for the project.

5. **Software development tools** [R, No default, Required] - level of software development tools availability ranging from basic debuggers and full screen editors to sophisticated program code generators that greatly enhance productivity.

6. **Technology impacts** [R, Default = Average, Required] - extent to which advances in technology will lead to a modification of system or software requirements during the development.

7. **COTS software** [R, Default = Average, Required] - level of effort required to integrate off-the-shelf software with the operational software that is being developed.

Computer Attributes

1. **Hardware system type** [R, No default, Required] - hardware configuration of the system on which the software will be hosted.

2. **Core utilized** [%, No default, Required] - amount of machine internal storage that is utilized by the operational software.

3. **Workstation types** [#, No default, Required] - number of workstation types requiring special screen clearing and set-up operations.

4. **Microprocessor code** [%, Default = Average, Required]- percentage of the software functions (with respect to the total software job) that are to be firmware.

5. **Hardware constraints** [R, Default = Average, Required] - degree to which the software system utilizes available processor memory, speed, and throughput.

6. **Development versus host system** [R, Default = Average, Required] -significance of differences between the development hardware system and the host system.

Personnel Attributes

1. **Hardware experience** [R, Default = Average, Required]- company's experience with the hosting development hardware equipment.

2. **Software experience** [R, Default = Average, Required]- company's experience with the software language and operating system.

3. **Development team** [R, Default = Average, Required]- development team's level of familiarity with the type of function to be developed.

Environmental Factors

1. **Development locations** [#, No default, Required] - the number of locations at which the software is to be developed.

2. **Customer locations** [#, No default, Required] - number of customer locations requiring the transfer of software development information or reviews during the software development cycle.

3. **Security level** [R, No default, Required] - level of computer security requirements that control access to system code, data, and software functions.

4. **Information travel requirements** [R, Default = Average, Required] - amount of information travel required between customer and contractor locations.

5. **Personnel resources** [R, No default, Required] - relative difficulty for staffing of the project due to special training and security requirements of project personnel.

6. **Development facilities** [R, Default = average, Required]- address the characteristics of the development facilities and the perceived availability of computer hardware.

## SoftCost-Ada

Product Attributes

1. **Type of software** [C, Default = other, Required] - application specific domain of software to be developed.

2. **Ada usage factor** [R, Default = nominal, Required]- percentage of the software being developed that will be written in the Ada programming language.

3. **Product complexity** [R, Default = nominal, Required] - level of complexity of the module to be developed as a function of the processing logic, optimization and efficiency concerns.

4. **Degree of real-time** [R, Default = nominal, Required] - the amount of tasking required to meet the system's performance requirements.

5. **Degree of reuse** [R, Default = nominal, Required] - percentage of software that will be packaged for reuse on other applications.

Sizing Factors

1. **Database size** [R, Default = nominal, Required] - relative database size (in bytes), represented as a percentage of the total program size.

2. **Subprojects** [#, No default, Required] - number of subprojects/deliverables/CSCI's that are defined to be part of the project. Subprojects are defined if a separate estimate is to be developed for each of them.

Note: The project may be sized using either a function point count or source lines of code count (SLOC).

Items 3 through 7 are required for function point sizing

3. **Function Point Count** [#, No default, Required] - raw function point count calculated for the software system based on the number of external inputs, outputs, inquiries, interfaces, internal files, operating modes, rendezvous, and stimulus/response relationships.

4. **Operand/Operator Count** [#, No default, Required] - measure of the size consumed by the mathematical equations specified for the system.

5. **Language type** [C, Default = Ada, Required] programming language which will be used to develop the software.

6. **Architectural factor** [E, Default = 1.00, Required]- adjustment which accounts for the impact of alternative system architectures. The value is calculated by the ASSET-R package.

7. **Technology factor** [E, Default = 1.00, Required] - adjustment which accounts for the experience base and capabilities of the software development organization. The value is calculated by the ASSET-R package.

Note: Items 8 through 13 are required for SLOC sizing

8. **Ada components size (new)** [#, No default, Required] - lines of new Ada code to be developed by this project including instantiations and program specification.

9. **Ada components size (reused)** [#, No default, Required]- lines of Ada code reused as-is, without modification. Each instantiated component should be specified once, regardless on the number of times it will be invoked.

10. **Ada components size (modified)** [#, No default, Required]- lines of existing Ada code reused as an Ada component, but modified during incorporation into the software system.

11. **Other components size (new)** [#, No default, Required] - lines of new code, written in a language other than Ada, that will be incorporated as part of the software system.

12. **Other components size (reused)** [#, No default, Required]- lines of code written in a language other than Ada, reused as a component and incorporated into the software system, without modification.

13. **Other components size (modified)** [#, No default, Required]- lines of code written in a language other than Ada, reused as a component but modified during incorporation into the software system.

Process Attributes

1. **Required development schedule** [R, Default = nominal, Required] - level of schedule constraint imposed on the project team developing a software subsystem expressed in terms of the percentage of schedule stretchout or acceleration with respect to a nominal schedule with nominal effort.

2. **Degree of standardization** [R, Default = nominal, Required]- degree to which language, life cycle, and military standards are applied to the software development effort.

3. **Scope of support** [R, Default = nominal, Required] - level of support that the developer will provide to non-software organizations.

4. **Use of modern software methods** [R, Default = nominal, Required] - type of software methods, such as structured programming or object-oriented design, that will be used during the software development effort.

5. **Use of peer reviews** [R, Default = nominal, Required] - the types of reviews that will be used during the software development effort.

6. **Use of software tools/environments** [R, Default = nominal, Required] - level of software tools and tool environments used in the development process ranging from basic Ada language tools to a full integrated life cycle APSE.

7. **Requirements volatility** [R, Default = nominal, Required]- percentage of requirements that are well established and will not change.

Computer Attributes

1. **System architecture** [C, Default = centralized architecture using a single processor, Required] - hardware configuration of the system on which the software will be hosted.

2. **Software tools/environment stability** [R, Default = nominal, Required] - stability and capability of the compiler and the frequency of changes to the software tools and tools environments used in development.

3. **Degree of optimization** [R, Default = nominal, Required]- degree to which available processor memory, speed, and input/output are utilized.

Personnel Attributes

1. **Ada projects completed** [#, No default, Required] - combined average of the number of Ada projects completed by members of the project team.

2. **Analyst capability** [R, Default = nominal, Required] - level of capability of the analysts expressed in terms of percentiles with respect to the overall population of analysts.

3. **Applications experience** [R, Default = nominal, Required]- average experience the analysts have with the types of applications within the software system.

4. **Ada environment experience** [R, Default = nominal, Required]- average experience the analysts have with the Ada software development environment that will be used during development of the software.

5. **Ada language experience** [R, Default = nominal, Required]- average experience the analysts have with the Ada programming language.

6. **Ada methodology experience** [R, Default = nominal, Required]- average experience the analysts have with the chosen Ada development methodology.

7. **Team capability** [R, Default = nominal, Required] - type of team - design, programming, or interdisciplinary, to be used for the software development effort.

Environmental Factors

1. **Software organizations involved** [#, No default, Required]- organizations which provide level of effort support (i.e. configuration management and quality assurance) and system level support.

2. **Organizational interface complexity** [R, Default = nominal, Required] - number of internal and external interfaces between all involved personnel and the degree to which organizations involved in the effort are geographically dispersed.

3. **Security requirements** [R, Default = nominal, Required]- security measures that must surround the development effort.

4. **Resource availability** [R, Default = nominal, Required]- percentage of staff availability assigned to the project and accessibility to software tools and equipment.

SPQR/20

Product Attributes

1. **Estimate type** [C, No default, Required] - indicates whether the project being estimated is a new program, an enhancement, or a maintenance change.

2. **Estimate scope** [C, No default, Required] - the kind of program or system that the estimate is for: prototype, module of a program, reusable module for multiple programs, complete stand-alone program, system, release, etc.

3. **Project class** [C, No default, Required] - indicates whether the software is to be used internally or externally, at single or multiple locations, developed under commercial, government, or military contract, and used by others in the public domain or sold as a retail product.

4. **Project type** [C, No default, Required] - application domain of the software to be developed.

5. **New code language** [C, No default, Required] - source language in which new software is to be developed.

6. **New code language level** [R, Default, Required] - the number of assembly language statements it will take to create the same function that one statement will take in the **new code language**.

7. **New code logical complexity** [R, Default = Average, Required] - rating from 1 (mostly simple algorithms) to 5 (many complex calculations) for level of complexity of the problem or algorithms to be coded.

8. **New code structural complexity** [R, Default = Well Structured, Required] -rating from 1 (nonprocedural) to 5 (poor structure with many complex paths and modules) for new program structure.

9. **New code data complexity** [R, Default = Multiple Files, Fields, and Data Interactions, Required] - rating from 1 (simple data with few variables) to 5 (very complex data elements and data interaction) for complexity of both the file structure and the data elements which the program or system will utilize.

10. **Base code language** [C, No default, Required for enhancement or maintenance **estimate type**] - refers to existing code of a program or system.

11. **Base code language level** [R, Default, Required for enhancement or maintenance **estimate type**] - the number of assembly language statements it will take to create the same function that one statement will take in the **base code language**.

12. **Base logical complexity** [R, Default = Average, Required for enhancement or maintenance **estimate type**] - rating from 1 (mostly simple algorithms) to 5 (many complex calculations) for level of complexity of the problem or algorithms of the base system.

13. **Base code structural complexity** [R, Default = Well Structured, Required for enhancement or maintenance **estimate type**] - rating from 1 (nonprocedural) to 5 (poor structure with many complex paths and modules) for base program structure.

14. **Base code data complexity** [R, Default = Multiple Files, Switches, and Data Interactions, Required for enhancement or maintenance **estimate type**] - rating from 1 (simple data with few variables) to 5 (very complex data elements and data interaction) for complexity of both the file structure and the data elements which the base program or system utilizes.

15. **Reusable code language** [C, No default, Required if **source code reusability** > 0%] - refers to code that will be reused from an existing application.

16. **Reusable code language level** [R, default, Required if **source code reusability** > 0%] - the number of assembly language statements it will take to create the same function that one statement will take in the **reusable code language**.

Sizing Factors

1. **Source code reusability** [R, Default = Moderate (> 25%), Required] - rating from 1 (extensive, > 75% use of reusable code) to 5 (no use of reusable code) for the amount of included code from software libraries.

2. **Reusable code size** [#, No default, Required if **source code reusability** > 0%] - refers to reused source code.

3. **Base code size** [#, No default, Required for enhancement or maintenance **estimate type**] - refers to existing code of a program or system.

NOTE: SPQR/20 will optionally estimate new code size using the function point methodology if a SLOC value is not entered directly for new code size.

4. **New code size** [#, No default, Required for direct sizing]- refers to new code to be developed. The line of code count includes procedural statements and data definitions and excludes comments, JCL, and included code.

NOTE: Items 5 through 9 are required inputs for function point sizing

5. **Inputs** [#, No default, Required for function point sizing]- unique number of input file, control information, or input screens that enter a program from an external source. Inputs are counted if they require unique processing logic or introduce new formats.

6. **Outputs** [#, No default, Required for function point sizing]- unique number of output files, control information, or output screens that leave a program and go to an external source. Outputs are counted if they require unique processing logic or introduce new formats.

7. **Inquiries** [#, No default, Required for function point sizing] - unique input/output combination such as a HELP screen, selection menu, or inquiry where an input is entered to direct a search of internal files and generate an immediate output. Inquires are counted if they require unique processing logic and cause no change to internal data.

8. **Data files** [#, No default, Required for function point sizing] - number of input and output files which the program will generate, access, or update. Also, each hierarchical path through a database or table in a relational database that requires unique processing.

9. **Interfaces** [#, No default, Required for function point sizing] - files or databases passed between or shared among separate applications.

Process Attributes

1. **Project estimating goals** [C, No default, Required] - user imposed constraints to be factored into the SPQR/20 estimate relating to staff size, schedule, cost, and required quality and reliability.

2. **Requirements definition** [R, Default = Fairly Clear User Requirements, Required] - rating from 1 (program developers are also users of the program) to 5 (uncertain, changing, ambiguous requirements) indicating the level of clarity and completeness of requirements definition.

3. **Program design automation** [C, Default = New Designs with Partial Text/Graphics Support, Required] - rating from 1 (reusable designs with automated text/graphics support) to 5 (new designs or hasty design with no automation) signifying the level of program design automation.

4. **User documentation** [C, Default = Programmers or Users; Automated Graphics/Text Support, Required] - identifies whether professional writers, programmers or users write documentation and the amount of documentation automation utilized.

Computer Attributes

1. **Terminal response time** [C, Default = 1-5 Seconds, Required]- average response time in seconds from the time the Return (Enter) key is depressed until the terminal responds.

Personnel Attributes

Note: Maximum and minimum staff size are selected by the user if **project estimating goals** stipulate constrained staffing.

1. **Maximum staff size** [#, No default, optional] - user imposed staff size constraint to be factored into the SPQR/20 estimate.

2. **Minimum staff size** [#, No default, optional] - user imposed staff size constraint to be factored into the SPQR/20 estimate.

3. **Staff availability** [%, Default = 100%, Required] - average time of availability of project personnel to be assigned to the project.

4. **Average work week** [#, Default = 40 hours, Required] - average work week for the staff on the project being estimated.

5. **Average work year** [#, Default = 220 days, Required] - average number of days in a work year for the organization.

6. **Staff experience** [R, Default = Even Mixture of Experts and New Hires or Novices, Required] - ratio of experienced project personnel familiar with the program to new hires and novices new the application.

Environmental Factors

1. **Project novelty** [C, Default = Even Mixture of New and Repeated Features, Required] - novelty of application to the organization.

2. **Office facilities** [C, Default = Multi-Employee Offices, Required] - physical environment where software development takes place such as individual offices with excellent facilities or an open office arrangement.

## SYSTEM-3

Product Attributes

1. **Application complexity** [R, Default = nominal, Required]- inherent difficulty associated with the application under estimation considering the number of years of work or study required to become proficient in the application area.

2. **Display requirements** [R, Default = nominal, Required]- amount of extra effort required to interface with the user.

3. **Language type** [R, Default = nominal, Required] - inherent complexity, magnitude, and learning difficulty of the development programming language.

4. **Real time operation** [R, Default = nominal, Required]- percentage of the development task that must perform its processing function within absolute time constraints.

5. **Time constraint** [R, Default = nominal, Required] - percentage of code that must have special attention to ensure adequate time performance (not real time) in order to meet overall system performance requirements.

Sizing Factors

1. **Source lines of code** [#, No default, Required] - amount of programmer developed source lines of code including all executable source lines, data declarations, and format statements and excluding comments and continuations.

Process Attributes

1. **Modern development practices** [R, Default = nominal, Required] - degree to which modern development practices are used in developing the software.

2. **Quality assurance level** [R, Default = nominal, Required]- required quality assurance effort based on the impact of a software failure.

3. **Requirements volatility** [R, Default = nominal, Required]- frequency and criticality of anticipated changes to the software requirements specification during development.

4. **Specification level** [R, Default = nominal, Required] - level of documentation that will be produced during the project development usually consistent with **quality assurance level** and **test level**.

5. **Test level** [R, Default = nominal, Required] - depth to which the software will undergo testing through Final Qualification Test usually based on the impact of a software failure during operation.

6. **Tool support, automated** [R, Default = nominal, Required]- availability and use of automated tools by the development team ranging from primitive to the advanced APSE.

Computer Attributes

1. **Memory constraints** [R, Default = nominal, Required] - effort required to make the software function within available computer memory.

2. **Rehosting effort** [R, Default = nominal, Required] - effort required to convert the software from the development facilities to the target system based on the extent of language and system changes required.

3. **Resource dedication** [R, Default = nominal, Required]- percentage of availability of the virtual machine to the software development project.

4. **Virtual machine complexity** [R, Default = nominal, Required]- relative difficulty to fully understand all virtual machine features.

5. **Turnaround - Logon through hardcopy** [R, Default = nominal, Required] - average time required to log onto the system, perform command actions to edit and save a file, compile, link, execute, print a source listing, and receive a hardcopy.

6. **Virtual machine volatility** [R, Default = nominal, Required] - frequency and criticality (major/minor) of changes to the virtual machine during development.

Personnel Attributes

1. **Analyst capability** [R, Default = nominal, Required] - level of capability of the analysts rated according to team efficiency, motivation, attitude, quality of previous work, ability to communicate, and ability to learn quickly.

2. **Analyst applications experience** [R, Default = nominal, Required] - the project team's average level of experience with similar applications.

3. **Language experience** [R, Default = nominal, Required] - the project team's average level of experience with the programming language to be used.

4. **Programmer capability** [R, Default = nominal, Required]- level of capability of the programmers based on ability to define design details, develop code, prepare test cases for program modules, and integrate modules.

5. **Virtual machine experience** [R, Default = nominal, Required]- the project team's average level of experience with the underlying virtual machine for an application.

Environmental Factors

1. **Multiple site development** [R, Default = nominal, Required]- additional time required for development due to the number and location of different development organizations.

2. **Resource location** [R, Default = nominal, Required] - degree of access to development resources, system consultants, programming language support, and software tools support.

This page is intentionally left blank.

## APPENDIX C

## HARDWARE REQUIREMENTS

Table C-1 summarizes the hardware requirements for each of the models. All of the models are available on an IBM PC (or compatible). Additional details concerning hardware requirements are provided in the following text.

TABLE C-1

HARDWARE REQUIREMENTS

| | IBM PC | ZENITH-248 | PRIME | VAX | MODEM |
|---|---|---|---|---|---|
| COSTMODL | X | | | | |
| PRICE S | X | | X | | X |
| SASET | X | | | | |
| SoftCost-Ada | X | | | X | |
| SPQR/20 | X | | | | |
| SYSTEM-3 | X | X | | | |

X = Available to DoD and Commercial users

1. **COSTMODL:** COSTMODL runs on IBM PCs and compatibles. A hard disk and 640K bytes of memory are required. Any monitor may be used, but a color monitor is preferred since color is used to differentiate between different classes of data.

2. **PRICE S:** PRICE S runs on a PRIME minicomputer operating under PRIMOS. In addition, PRICE S can be accessed via a time-sharing system with an office terminal and standard modem.

3. **SASET:** SASET may be hosted on any IBM PC or compatible with a minimum of 512K bytes of memory, one disk drive, and an 8088/86, 80186, 80286, 80386 microprocessor running PC-DOS or MS-DOS, version 2.0 or higher. The model functions with either a color or monochrome monitor. A hard disk and printer are optional.

4. **SoftCost-Ada:** SoftCost-Ada runs on an IBM PC, PC/XT, PC/AT, PS/2 or compatible with a minimum of 256K bytes of memory and a color or monochrome display. The system requires PC-DOS or MS-DOS, version 2.0 or higher. A minimum of one floppy disk drive is required. A hard disk drive and printer are optional. SoftCost-Ada may also be hosted on the Digital MicroVax II or VAX 11/780 with VMS version 4.6 or higher.

5. **SPQR/20:** SPQR/20 runs on an IBM PC, XT, AT, or compatible with 512K bytes of memory and a color or monochrome display. Two floppy disk drives or a floppy disk drive and a hard disk drive are required.

6. **SYSTEM-3:** SYSTEM-3 runs on an IBM PC, XT, AT, Zenith 248 or compatibles with a minimum of 512K bytes of memory and a color or monochrome display. The system requires PC-DOS or MS-DOS, version 2.0 or higher. A minimum or one floppy disk drive is required.

## APPENDIX D

## CONTRACTUAL ARRANGEMENTS AND COSTS

Tables D-1 and D-2 summarize the availability and cost of models applied in the test case study. Table D-1 summarizes the availability of each model to DoD and commercial users. Availability through request means that a potential user can receive the model at no cost by contacting the model POC. The model is received on diskettes that are provided by the requesting agency. Table D-2 shows the DoD rates for each of the models. Separate commercial rates apply to PRICE S and System-3. Additional costs may be associated with user training, which is required for some of the models. Also, rates will vary depending upon the type of licensing agreement procured (annual, site, corporate, etc.).

1. **COSTMODL:** COSTMODL is available to all DoD and commercial users. Version 5.0 of COSTMODL is available by contacting Bernie Roush at NASA- Johnson Space Center. Version 5.0 implements the complete Ada COCOMO model and the Incremental Development model which were intoduced by Dr. Boehm at the November 1987 COCOMO User's Group Conference. It does not include the enhancements to the Ada model that Dr. Boehm introduced at the 1988 COCOMO User's Group Conference. Enhancements will be incorporated in Version 6.0. Requests should be accompanied with three 360K 5.25" disks. Implementors are currently soliciting feedback on the package's users-interface. After upgrades, COSTMODL will be available from

   NASA/COSMIC
   The University of Georgia
   Computer Services Annex
   Athens, GA 30602
   (404) 542-3265

   There is a nominal handling charge for the program.

2. **PRICE S:** PRICE S is part of the PRICE system of models that includes PRICE SZ for software sizing, and PRICE SL for software life cycle costs (maintenance, enhancement, and growth activities associated with life-cycle support). Government users can use the PRICE S package on a time sharing basis at $82 per hour (through 11/91) by contacting Lt. Ken Nelson of the Aeronautical Systems Division at Wright Patterson AFB. Commercial users can use the PRICE S package on a time sharing basis at $15/hour of connect time and

TABLE D-1

CONTRACTUAL ARRANGEMENTS

| | PURCHASE | LEASE | TIME SHARE | REQUEST |
|---|---|---|---|---|
| COSTMODL | | | | X |
| PRICE S | | X | X | |
| SASET | | | | D |
| SoftCost-Ada | | X | | |
| SPQR/20 | X | | | |
| SYSTEM-3 | | X | | |

TABLE D-2

LEASE/PURCHASE RATES
(DoD)

| | FIRST UNIT | EXTRA UNITS | TIME SHARE |
|---|---|---|---|
| COSTMODL | No Cost | | |
| PRICE S | | | $82/Hour |
| SASET | No cost, but controlled access | | |
| SoftCost-Ada | $8,000 | $1,000/Copy | |
| SPQR/20 | $5,000 | Negotiable | |
| SYSTEM-3 | $9,550/Year | $800/Copy | |

$0.060/resource unit of CPU time by contacting PRICE Systems at Moorestown, New Jersey, but, they must also pay an access fee of $40,000/year for one unit or $60,000/year for unlimited access. Commercial users can also lease the PRICE S package for installation on their own PRIME minicomputer at $60,000/year for one user at a time or $80,000/year for unlimited access. A one week training course is mandatory and costs $1,312.50 (through 11/91) for a government student or $1,750 for a commercial student. These costs include refresher training, manual updates, technical assistance, and newsletter at no additional charge.

3. **SASET:** SASET is presently being used by the U.S. Air Force Cost Center and the Naval Center for Cost Analysis. Availability to DoD users on a broader basis is an issue that will be decided by Mr. Steve Gross and the Naval Center for Cost Analysis. There are presently no plans to market the SASET model, but Martin Marietta Corporation may market a derivative model.

4. **SoftCost-Ada:** SoftCost-Ada is available for a monthly or annual licensing fee. The SoftCost-Ada PC version annual licensing fee for one unit costs $8,000. The price for additional copies is $1,000. A site license is $11,000. The SoftCost-Ada Vax version costs $8,000 for the first license (4 users) and $1,000 for each additional license (4 users). Prices include a telephone help line and system upgrades at no additional charge. SoftCost-Ada Vax version site licensing agreements are negotiable. A GSA contract is being negotiated which will result in a discount for DoD users.

5. **SPQR/20:** A SPQR/20 one time licensing fee for purchasing one unit costs $5,000. Site licenses and multi-volume purchases are negotiable.

6. **SYSTEM-3:** The System-3 annual rate for government users is $9,550/year for one unit; additional units (two and three) are $800/year and $600/year for four or more units. The System-3 annual licensing fee for commercial users is $12,500 for one unit; additional units (two through four) are $2,000/year. In addition, further price reductions are available for blocks of five, ten, 25 and 50 units. System-3 has a training course available. This course is strongly recommended and costs $790/person when given at the CEI facility. Training at the customer's facility (up to 20 persons per session) is $4500/session plus travel and living expenses for CEI personnel. Commercial training costs are $5,800/per session plus travel and living expenses. Prices include a telephone help line and system upgrades at no additional charge.

This page is intentionally left blank.

# APPENDIX E

## SCOPE OF COVERAGE: LIFE CYCLE PHASES AND ACTIVITIES

With the exception of SPQR/20, each model included in this Ada costing study generates an effort expenditure summary in terms of the software cost elements encompassed by the estimate and by life cycle phase. SPQR/20 provides estimates only in terms of the software project activity. However, these activities can be mapped to life-cycle phases. Table E-2 shows the range of life-cycle phases covered by each model. Phases are mapped to technical reviews and audits [DOD-STD-2167A] in order to provide a basis for comparison of models in terms of life-cycle coverage. Blocks depicted in Table E-1 are labelled using the same phase terminology prescribed by each model. It is evident from the Table that phases are not defined in a standard way across all models. All of the models cover the operational or maintenance phase in addition to development. Operational support, following successful completion of a software acceptance review (FQR), estimated for each model is provided in Table E-1.

A breakdown of software cost elements encompassed by the model estimates is provided in Table E-2. Activities are described using the exact terminology of the model. A separate estimate given in terms of person-months of effort is provided for each cost element.

TABLE E-1

OPERATIONAL SUPPORT ACTIVITIES

| | |
|---|---|
| COSTMODL: | Annual maintenance |
| PRICE S: | Operational support for user-specified length |
| SASET: | Operational support for user-specified length |
| SoftCost-Ada: | Operational support for user-specified length |
| SPQR/20: | Up to 5 years of operational support |
| SYSTEM-3: | 15 years of operational support |

TABLE E-2

SCOPE OF COVERAGE: LIFE CYCLE PHASES

| | C/A – SRR | SRR – SDR | SDR – SSR | SSR – PDR | PDR – CDR | CDR – TRR | TRR – PCA | PCA – FQR | FQR |
|---|---|---|---|---|---|---|---|---|---|
| COSTMODL | | | *Plans & Reqts | Product Design | Programming | Programming | Integ. and Test | Integ. and Test | Maint. |
| PRICE S | System Concept | Sys/Software Requirements | Software Requirements | Preliminary Design | Detail Design | Code/ Test | CSCI Test | System Integration and Test* | Oper T&E |
| SASET | System Reqts | Requirements Allocation | Software Requirements | Preliminary Design | Detail Design | Code / Check Out | Unit Test / PQT/ FQT Integ | System Test and Integ* | Maint. |
| SOFTCOST ADA | System Definition | System Definition | Software Requirements | Architectural Design | Detail Design | Implemt. | Software Testing | System Testing | O&S |
| SPQR/20 | Planning | Planning | Requirements | Design | Design | Code | Integration & Test | Integration & Test | Maint. |
| SYSTEM-3 | C/A - SDR | C/A - SDR | SDR - PDR | SDR - PDR | PDR-CDR | CDR-CUT | CUT-FQT | CUT-FQT | DT&E |

* 8% of Total Effort Estimated by the Model

* Resources expended during this phase are not included in the case study.

TABLE E-3

SOFTWARE COST ELEMENTS ENCOMPASSED BY MODEL ESTIMATES

| MODEL | ACTIVITY |
|---|---|
| COSTMODL | . Requirements Analyses<br>. Product Design<br>. Programming<br>. Test Planning<br>. Verification and Validation<br>. Project Office<br>. Configuration Management/Quality Assurance<br>. Manuals |
| PRICE S | . Software Design<br>. Programming<br>. Documentation<br>. Systems Engineering and Program Management<br>. Quality Assurance<br>. Configuration Management |
| SASET | . Software Engineering<br>. Systems Engineering<br>. Quality Assurance<br>. Test Engineering |
| SoftCost-Ada | . Software Development<br>. Software Management<br>. Software Configuration Management<br>. Software Quality Evaluation |
| SPQR/20 | . Planning<br>. Requirements<br>. Design<br>. Coding<br>. Integration/Test<br>. Documentation<br>. Management |
| SYSTEM-3 | . Systems Engineering<br>. Project Management<br>. Design<br>. Programmers<br>. Quality Assurance<br>. Configuration Management<br>. Test<br>. Data Manipulation |

This page is intentionally left blank.

## APPENDIX F

### Detailed Description of the Ada COCOMO Model

Appendix F provides a description of the Ada COCOMO Model which was introduced by Dr. Boehm at the 1987 COCOMO User's Group Conference. This description includes the enhancements to the Ada model, namely the interpolation of ACAP, PCAP, and MODP multipliers, that Dr. Boehm introduced at the 1988 COCOMO User's Group Conference. Note that current estimating equations only apply to embedded-mode software projects (see pages 78-83, Software Engineering Economics). Equations have not been developed for organic or semidetached modes, or Basic COCOMO [BOEH88]. This overview is organized into the following sections:

I. Estimating Nominal Effort

A. Accessing Compliance with the Ada Process Model
B. Ada Instruction Counting Rules

II. Adjust Nominal Effort

A. Rate Software Development Effort Multipliers
B. Interpolate ACAP, PCAP and MODP Multipliers
C. Apply Effort Multiplier

III. Sample Problem

### I. Estimating Nominal Effort

An Ada COCOMO software development effort estimate begins by determining the size of the effort and the extent to which the development effort complies with the Ada Process Model. Once these values have been determined, a nominal effort estimate is generated, using the following equation:

$$MM_{nom} = 2.8\ (KDSI)^{1.04\ +\ b} \qquad (1)$$

where

| | | |
|---|---|---|
| KDSI | = | Thousands of delivered source instructions |
| b | = | extent to which the development effort complies with the Ada Process Model. |

The values that b can assume range from b = 0, for full use of the Ada Process Model, to b = 0.20, for non-use of the Ada Process Model. Since the extent of compliance with the Ada Process Model appears in the equation as an exponent, it is very important to make intelligent selections for each of the Ada Process Model compliance characteristics.

## Assessing Compliance with the Ada Process Model

Table F-1 describes the characteristics which determine the extent of compliance with the Ada Process Model (APM).

TABLE F-1

CHARACTERISTICS THAT DETERMINE EXTENT OF COMPLIANCE WITH APM

| WEIGHTING VARIABLE | CHARACTERISTIC | RATING SCALE | | | | | |
|---|---|---|---|---|---|---|---|
| | | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |
| $b_1$ | Experience with Ada Process Model | Greatest Experience | Greater Experience | General Experience | Some Experience | Little Experience | No Experience |
| $b_2$ | Design at PDR, Unit Package Spec Compiled, Body Outlined | 100 % Completed | 90 % Completed | 75 % Completed | 60 % Completed | 40 % Completed | <= 20 % Completed |
| $b_3$ | Risks Eliminated by PDR | 100 % Eliminated | 90 % Eliminated | 75 % Eliminated | 60 % Eliminated | 40 % Eliminated | <= 20 % Eliminated |
| $b_4$ | Requirements Volatility During Development | No Changes | Small Non-Critical | Frequent Non-Critical | Occasional Moderate | Frequent Moderate | Many Large |

The sum of the individual ratings, b, is used in the exponent portion of the nominal effort equation (1). Tables F-2, F-3, and F-4 are provided to aid in this selection process for determining Design Thoroughness at PDR, Risk Elimination by PDR, and Requirements Volatility. To select the appropriate Ada Process Model compliance rating, one averages the rates of the respective support criteria.

TABLE F-2

**ADA COCOMO b FACTOR : DESIGN THOROUGHNESS BY PDR**

| CHARACTERISTIC \| $b_i$ RATING | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |
|---|---|---|---|---|---|---|
| * Schedule, budget, and internal milestones through PDR compatible with Risk Management Plan | Fully | Mostly | Generally | Some | Little | None |
| * Percent of development schedule devoted to Preliminary Design Phase | 40 % | 33 % | 25 % | 17 % | 10 % | 5 % |
| * Percent of required top software architects available to project | 120 % | 100 % | 80 % | 60 % | 40 % | 20 % |
| * Tool support for developing and verifying Ada package specs | Full | Strong | Good | Some | Little | None |
| * Level of uncertainty in key architecture drivers: mission, user interface, hardware, technology | Very Little | Little | Some | Considerable | Significant | Extreme |

TABLE F-3

**ADA COCOMO b FACTOR : RISK ELIMINATION BY PDR**

| \ $b_i$ RATING<br>CHARACTERISTIC \ | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |
|---|---|---|---|---|---|---|
| * Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR | Fully | Mostly | Generally | Some | Little | None |
| * Schedule, budget, and internal milestones through PDR compatible with Risk Management Plan | Fully | Mostly | Generally | Some | Little | None |
| * Percent of development schedule devoted to Preliminary Design Phase | 40 % | 33 % | 25 % | 17 % | 10 % | 5 % |
| * Percent of required top software architects available to project | 120 % | 100 % | 80 % | 60 % | 40 % | 20 % |
| * Tool support for resolving risk items | Full | Strong | Good | Some | Little | None |

TABLE F-4

ADA COCOMO b FACTOR : REQUIREMENTS VOLATILITY

| \ $b_i$ RATING<br>CHARACTERISTIC \ | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 |
|---|---|---|---|---|---|---|
| * System requirements baselined, under rigorous change control | Fully | Mostly | Generally | Some | Little | None |
| * Level of uncertainty in key requirements areas: mission, user interface, hardware, other interfaces | Very Little | Little | Some | Considerable | Significant | Extreme |
| * Organizational track record in keeping requirements stable | Excellent | Strong | Good | Moderate | Weak | Very Weak |
| * Use of incremental development to stabilize requirements | Full | Strong | Good | Some | Little | None |
| * System architecture modularized around major sources of change | Fully | Mostly | Generally | Some | Little | None |

## Ada Instruction Counting Rules

In order to complete the nominal effort estimate it is necessary to determine the number of Ada instructions present in the software development effort. The following criteria are used to determine what an Ada instruction is:

1) Delivered Source Instructions =
   Number of Carriage Returns in Package Specs
   + Number of Semicolons in Package Bodies

2) Comments are not counted

3) Reuse and multiply-used packages are counted as follows:

   a) Unmodified utility software is not counted
   b) Count each invocation statement ( e.g., WITH, USE )
   c) Use the Reuse Model to count packages invoked:

$$EDSI = (ADSI)\ [0.4\ (DM) + 0.3\ (CM) + 0.3\ (IM)]\ /\ 100$$

| | | |
|---|---|---|
| EDSI | = | Equivalent new DSI |
| ADSI | = | Adapted or reused instructions |
| DM | = | % of adapted software design modified |
| CM | = | % of adapted code modified |
| IM | = | % of adapted software's original integration required in new context. |

## II. Adjust Nominal Effort

Once a nominal effort estimate ($MM_{nom}$) has been generated, it is adjusted by applying effort multipliers determined from the project's ratings with respect to 18 cost driver attributes. The product of all 18 cost driver attributes, EM, and the nominal effort estimate, $MM_{nom}$, yield man-months for development.

$$MM = MM_{nom} * \prod_{i=1}^{18} (EM) \qquad (2)$$

### Rate Software Development Effort Multipliers

The numerical values of the effort multipliers (EM) are listed in Table F-5. The corresponding rating scale is summarized for each effort multiplier:

1) Required Software Reliability (RELY):

| | |
|---|---|
| VL | = Error effect, slight inconvenience |
| L | = Low, easily recoverable losses |
| NOM | = Moderate, recoverable losses |
| H | = High financial loss |
| VH | = Risk to human life |

2) Data Base Size (DATA):

| | |
|---|---|
| L | = DB bytes/program DSI < 10 |
| NOM | = 10 <= D/P < 100 |
| H | = 100 <= D/P < 1,000 |
| VH | = D/P >= 1,000 |

3) Software Product Complexity (CPLX):

See Table 8-4 in Software Engineering Economics, page 122.

4) Required reusability (RUSE). Ratings are assigned according to the extent that the developer must design, document, and test reusable Ada components:

| | |
|---|---|
| NOM | = Ada components not for reuse elsewhere |
| H | = Reuse within single-mission products |
| VH | = Reuse across single product line |
| EH | = Reused in any application |

5) Execution time constraint (TIME):

| | |
|---|---|
| NOM | = <= 50 % use of available execution time |
| H | = 70 % use |
| VH | = 85 % use |
| EH | = 95 % use |

6) Main storage constraint (STOR):

| | |
|---|---|
| NOM | = <= 50 % use of available execution time |
| H | = 70 % use |
| VH | = 85 % use |
| EH | = 95 % use |

7) Virtual machine volatility for host computer (VMVH). Ratings are defined in terms of the relative frequency of major changes and minor changes to the underlying virtual machine where

- o A major change significantly affects roughly 10 % of routines under development.

o A minor change significantly affects roughly 1 % of routines under development.

Ratings are assigned as follows:

| | |
|---|---|
| L | = Major: 12 months; Minor: 1 month |
| NOM | = Major: 6 months; Minor: 2 weeks |
| H | = Major: 2 months; Minor: 1 week |
| VH | = Major: 2 weeks; Minor: 2 days |

8) Virtual machine volatility for target computer (VMVT). The same definition for host effects applies to target effects.

Ratings are assigned as follows:

| | |
|---|---|
| L | = Major: 12 months; Minor: 1 month |
| NOM | = Major: 6 months; Minor: 2 weeks |
| H | = Major: 2 months; Minor: 1 week |
| VH | = Major: 2 weeks; Minor: 2 days |

9) Computer turnaround time (TURN). Ratings are defined in terms of average response time in hours (from the time the developer submits a job to be run until the results are back in the developer's hands):

| | |
|---|---|
| VL | = Interactive, 1 terminal/person, full lifecycle support |
| L | = Interactive, 0.3 terminals/person, focused on code, test support |
| NOM | = < 4 hours turnaround time |
| H | = 4 - 12 hours turnaround time |
| VH | = > 12 hours turnaround time |

10) Analyst capability (ACAP):

| | |
|---|---|
| VL | = 15th percentile team |
| L | = 35th percentile team |
| NOM | = 65th percentile team |
| H | = 75th percentile team |
| VH | = 90th percentile team |

11) Programmer capability (PCAP):

| | |
|---|---|
| VL | = 15th percentile team |
| L | = 35th percentile team |
| NOM | = 65th percentile team |
| H | = 75th percentile team |
| VH | = 90th percentile team |

12) Applications experience (AEXP):

| | |
|---|---|
| VL | = <= 4 months average experience |
| L | = 1 year average experience |
| NOM | = 3 years average experience |
| H | = 6 years average experience |
| VH | = >= 12 years average experience |

13) Virtual machine experience (VEXP):

| | |
|---|---|
| VL | = <= 1 month average experience |
| L | = 4 months average experience |
| NOM | = 1 year average experience |
| H | = >= 3 years average experience |

14) Programming language experience (LEXP). Duration of experience of the project team developing the software module with the programming language to be used:

| | |
|---|---|
| VL | = <= 1 month average experience |
| L | = 4 months average experience |
| NOM | = 1 year average experience |
| H | = 3 years average experience |
| VH | = >= 6 years average experience |

15) Use of modern programming developing practices (MODP):

| | |
|---|---|
| VL | = No use MODPs |
| L | = Beginning use |
| MOM | = Some use |
| H | = General use |
| VH | = Routine use |

16) Use of software tools (TOOL). Ratings are assigned according to the degree to which software tools are used in developing the software subsystem:

| | |
|---|---|
| VL | = Very few, primitive tools |
| L | = Basic micro tools |
| NOM | = Basic mini tools |
| H | = Basic maxi tools |
| VH | = Extensive tools, little integration |
| EH | = Moderately integrated environment (UNIX); strong target tool support |
| XXH | = Fully integrated environment; rating not acheived by any support environment to date, rating can be dropped for near term; strong target tool support |

17) Required development schedule (SCED):

| | |
|---|---|
| VL | = 75 % of nominal schedule |
| L | = 85 % |
| NOM | = 100 % |
| H | = 130 % |
| VH | = 160 % |

18) Classified security application (SECU). Ratings are assigned according to security and privacy restrictions:

| | |
|---|---|
| NOM | = Unclassified |
| H | = Classified (Secret, Top Secret) |

Interpolation of ACAP, PCAP, and MODP Effort Multipliers

Some of the effort multipliers are affected by the extent of Ada Process Model compliance: ACAP, PCAP, and MODP. To determine what the value of these modified effort multipliers should be, one must interpolate between numerical values for the full and non-use cases of Ada Process Model compliance. The ratio, b / 0.16, provides a measure to determine the distance between the value for the effort multiplier for Ada COCOMO and standard COCOMO. In the following example (Exhibit F-2), 0.08 is the value used for b, ACAP is set to low, PCAP to very low, and MODP to nominal. The effect of the interpolation of these effort multipliers is to extend their range.

TABLE F-6

ADA COCOMO b FACTOR: INTERPRETATION EFFECTS [BOEH88A]

For Effort Multipliers (EMs) ACAP, PCAP, and MODP

$n_{ew} = EM_{full} + b/0.16 \; ( EM_{non} - EM_{full})$

| COST DRIVER | VERY LOW | LOW | NOMINAL | HIGH | VERY HIGH |
|---|---|---|---|---|---|
| ACAP | 1.57 - 1.46 | 1.29 - 1.19 | 1.00 - 1.00 | 0.80 - 0.86 | 0.61 - 0.71 |
| PCAP | 1.30 - 1.42 | 1.12 - 1.17 | 1.00 - 1.00 | 0.89 - 0.86 | 0.80 - 0.70 |
| MODP | 1.24 - 1.24 | 1.10 - 1.10 | 0.98 - 1.00 | 0.86 - 0.91 | 0.78 - 0.82 |

$$ACAP_{new} = 1.29 + 0.08 / 0.16 \; ( 1.19 - 1.29 ) = 1.24$$

$$PCAP_{new} = 1.30 + 0.08 / 0.16 \; ( 1.42 - 1.30 ) = 1.36$$

$$MODP_{new} = 0.98 + 0.08 / 0.16 \; ( 1.00 - 0.98 ) = 0.99$$

Apply Effort Multiplier

Once the interpolation has been performed, the product of the 18 EMs and the nominal effort estimate, $MM_{nom}$, are used to adjust nominal effort (2). Once the adjusted affort is known (MM), the development schedule can be estimated. The following equation is employed to compute the development schedule estimate:

$$T_{dev} = 3.0 \; (MM)^{0.32 + (0.2 * b)}$$

where

$T_{dev}$ = Time in months from Software Requirements Review (SRR) to Software acceptance test (FQT)

III. Sample Problem

Let us run through an example, using one of the projects targeted in the test case study, Project 6. We start with the selections for extent of compliance with the Ada Process Model. The characteristic being evaluated, its rating and associated numerical value are provided in Table F-7.

TABLE F-7

ADA PROCESS MODEL CHARACTERISTICS FOR SAMPLE PROBLEM

| CHARACTERISTIC | RATING | VALUE |
|---|---|---|
| Experience with the APM | Some experience | 0.03 |
| Design thoroughness by PDR | 60 % eliminated | 0.03 |
| Risks Eliminated by PDR | 90 % eliminated | 0.01 |
| Requirements Volatility During Development | Frequent moderate changes | 0.04 |

The values of the ratings are summed:

$$b = 0.03 + 0.03 + 0.01 + 0.04 = 0.11$$

There were 24 K delivered source instructions (DSI), so a nominal effort estimate may be computed as follows:

$$MM_{nom} = 2.8\ (KDSI)^{1.04 + b} = 2.8\ (24)^{1.15} = 108.24$$

Since b = 0.11 represents partial compliance with the Ada Process Model, we interpolate to arrive at new effort multipliers for ACAP, PCAP, and MODP as below ( and in Exhibit F-4):

$$ACAP_{new} = 0.80 + 0.11 / 0.16\ (0.86 - 0.80) = 0.84125$$

$$PCAP_{new} = 0.89 + 0.11 / 0.16\ (0.86 - 0.89) = 0.869375$$

$$MODP_{new} = 0.98 + 0.11 / 0.16\ (1.00 - 0.98) = 0.99375$$

TABLE F-8

INTERPOLATED VALUES FOR SAMPLE PROBLEM

| Effort Multiplier | Rating | COCOMO Value Standard | Ada | b | Interpolated Value |
|---|---|---|---|---|---|
| **ACAP** | H | 0.86 | 0.80 | 0.11 | 0.84125 |
| **PCAP** | H | 0.86 | 0.89 | 0.11 | 0.869375 |
| **MODP** | NOM | 1.00 | 0.98 | 0.11 | 0.99375 |

Values for the effort multipliers are as follows:

| **RELY** | Value | **DATA** | Value | **CPLX** | Value | **RUSE** | Value |
|---|---|---|---|---|---|---|---|
| H | 1.07 | NOM | 1.00 | H | 1.08 | H | 1.10 |
| **TIME** | Value | **STOR** | Value | **VMVH** | Value | **VMVT** | Value |
| H | 1.11 | NOM | 1.00 | NOM | 1.00 | L | 0.93 |
| **TURN** | Value | **ACAP** | Value | **PCAP** | Value | **AEXP** | Value |
| VL | 0.79 | H | 0.84125 | H | 0.869375 | H | 0.91 |

| VEXP | Value | LEXP | Value | MODP | Value | TOOL | Value |
|---|---|---|---|---|---|---|---|
| VL | 1.21 | VL | 1.26 | NOM | 0.99375 | NOM | 1.00 |

| SCED | Value | SECU | Value |
|---|---|---|---|
| NOM | 1.00 | NOM | 1.00 |

The man-months for development, MM, can now be computed using the effort equation:

$$MM = MM_{nom} * \prod_{i=1}^{18} (EM)_i$$

Substituting, MM is calculated:

$$MM = 108.24 * 1.045 = 113.14$$

With man-months for development, MM, we can now generate the schedule estimate using the following relationship:

$$\begin{aligned} T_{dev} &= 3.0\ (113.14)^{0.32 + (0.2 * b)} \\ &= 3.0\ (113.14)^{0.342} = 15.12 \end{aligned}$$

So, for this project, effort multipliers extended the man-months for development from the nominal estimate of 108.24 to 113.14 man-months. The development schedule estimate was computed to be 15.12 months.

## APPENDIX G

## Project Questionnaire

This project questionnaire contains the data items that are required to run the following software cost and schedule estimating models:

- Ada COCOMO
- SoftCost-Ada
- SYSTEM-3
- SASET
- PRICE S
- ESTIMACS
- SPQR/20

An abbreviated version of this form was used to collect information on six of the nine projects targeted in the test case study. The form is modular and is divided into the following eight sections:

1. General Information
2. Project Description
3. Development Methodology
4. Software Size
5. Project Staffing
6. Computer System
7. Development Environment
8. Resource Allocation.

## PROJECT QUESTIONNAIRE

### GENERAL INFORMATION

Please complete this form to the best of your ability for your project. If the question is not applicable, please mark it N/A. If you don't know the answer, leave it blank. Mark each page containing confidential or proprietary data **"CONFIDENTIAL"** on both its top and bottom in bold letters.

1. Your name:______________________________ Date: / /
2. Title:______________________________ Phone: ( ) -
3. Firm or organization:______________________________

   Address:______________________________

   ______________________________

   ______________________________
4. Name of project:______________________________
5. Contract number:______________________________
6. Customer name:______________________________
7. Project overview description:______________________________

   ______________________________

   ______________________________

   ______________________________

   ______________________________

   ______________________________
8. Developer contact:______________________ Phone: ( ) -
9. Customer contact:______________________ Phone: ( ) -
10. Current status: ______________________________

# PROJECT QUESTIONNAIRE

## PROJECT DESCRIPTION

1. System/Software Characteristics

   a. Application Type (check all that apply):

      [ ] Prototype to be discarded later
      [ ] Prototype to be built into delivered program
      [ ] Complete stand-alone program
      [ ] Component within a system
      [ ] Reusable component for multiple programs
      [ ] Release with new features
      [ ] Release with defect repairs
      [ ] System with multiple components

   b. Project Class (check all that apply):

| | |
|---|---|
| [ ] Used at single location | [ ] Military contract |
| [ ] Used at multiple locations | [ ] Government contract |
| [ ] Sold as retail product | [ ] Commercial contract |
| [ ] For public domain | [ ] Leased to users |
| [ ] Bundled with hardware | [ ] For personal use |
| [ ] Developed in an Academic Environment | |
| [ ] Used remotely via time-sharing | |
| [ ] Using military specifications | |

   c. Applications domain (check all that are applicable):

| | |
|---|---|
| [ ] Automation | [ ] Data Processing |
| [ ] Business | [ ] Environments/Tools |
| [ ] Command & Control | [ ] Production Control |
| [ ] Communications | [ ] Support Software |
| [ ] Signal Processing | [ ] Process Control |
| [ ] Test Systems | [ ] Scientific |
| [ ] Trainers | [ ] Robotics/Mechanical |
| [ ] Avionics | [ ] Unmanned Flight |
| [ ] Interface Systems | [ ] Manned Flight |
| [ ] Graphics, image processing | |
| [ ] Other ________________________ | |

   d. Software Interface Complexity: How many software systems and peripheral communications equipment does this software system interface with? ______

## PROJECT QUESTIONNAIRE

<u>PROJECT DESCRIPTION</u>, cont'd.

e. Rate the difficulty of processing logic:

- [ ] Simple processing logic, straight-forward I/O
- [ ] Difficult highly nested logic, real-time processing
- [ ] Routine nesting, minimal interface with Operating system, standard I/O
- [ ] Complex dynamic resource allocation, multiple exception handlers, recursion

f. Product Structural Complexity

- [ ] Nonprocedural
- [ ] Well structured reusable modules
- [ ] Small modules and simple paths
- [ ] Complex modules and paths
- [ ] Large modules and complex paths

g. Mathematical Complexity

- [ ] Simple algorithms and simple calculations
- [ ] Majority of simple algorithms and calculations
- [ ] Algorithms and calculations of average complexity
- [ ] Some difficult or complex calculations
- [ ] Many difficult algorithms and complex calculations

h. Degree of Real-Time

- [ ] No tasking; essentially batch response
- [ ] Interactive with limited Ada tasking
- [ ] Interrupt driven with tasking in milliseconds
- [ ] Concurrent tasking with rendezvous in milliseconds
- [ ] Concurrent tasking with rendezvous in nanoseconds

i. List the percent of total source code allocated to each of the following operational timing requirements:

| | | | |
|---|---|---|---|
| Real-Time | _____% | On-line | _____% |
| Time-Constrained | _____% | Non-Time-Critical | _____% |

j. Select the percent of source code with real-time considerations:

[ ] 0% [ ] 25% [ ] 50% [ ] 100%

2. Database Characteristics

a. Number of physical data files:_____

## PROJECT QUESTIONNAIRE

**PROJECT DESCRIPTION**, cont'd.

b. Database Complexity

[ ] Simple data, few variables, low complexity
[ ] Simple, numerous variables
[ ] Multiple files, fields data interactions
[ ] Complex file structure
[ ] Highly complex

3. Reliability

a. Effect of a software failure

| | |
|---|---|
| [ ] Inconvenience | [ ] Moderate loss |
| [ ] Easily-recoverable loss | [ ] Major financial loss |
| [ ] Loss of human life | |

b. Backup/Recovery Considerations

[ ] Data protection beyond regular backup required
[ ] Alternative methods need to be developed in case of software failure
[ ] No special backup requirements

4. User Interface

a. Display Requirements

[ ] Simple I/O
[ ] User-friendly, menu driven
[ ] Pressure sensitive devices (touch screen, joystick)
[ ] Graphics oriented

5. Software Testability

| | |
|---|---|
| [ ] Very difficult | [ ] Time insensitive |
| [ ] Difficult | [ ] Easy |

6. Is this the first system of its kind?

[ ] Yes [ ] No

## PROJECT QUESTIONNAIRE

**PROJECT DESCRIPTION**, cont'd.

7. Reused Code

   a. Logical Complexity of Reused Code:

      [ ] Simple algorithms an dsimple calculations
      [ ] Majority of simple algorithms and calculations
      [ ] Algorithms and calculations of average complexity
      [ ] Some difficult or complex calculations
      [ ] Many difficult algorithms and complex calculations

   b. Structural Complexity of Reused Code:

      [ ] Nonporcedural (generated, query, spreadsheets, etc.)
      [ ] Well structured with reusable modules
      [ ] Well structured (small modules and simple paths)
      [ ] Fair structure but some complex paths and modules
      [ ] Poor structure with many complex paths and modules

   c. Database Complexity

      [ ] Simple data with few variables and little complexity
      [ ] Several data elements byt simple data relationships
      [ ] Multiple files, switches, and data interactions
      [ ] Complex data elements and domplex data interactins
      [ ] Very complex data elements and data interactions

   d. Select the intended use of the majority of the software packaged for reuse:

      [ ] None
      [ ] Reuse within single mission products
      [ ] Reuse across single product line
      [ ] Reuse in any application

8. General

   a. Scope of Support

      [ ] No support to non-software organizatoins
      [ ] Liaison support to non-software organizations
      [ ] Extensive support to system test organizations
      [ ] Extensive support to system entineering and test organizations. CSSR/CSCSC reporting requirements.

## PROJECT QUESTIONNAIRE

### DEVELOPMENT METHODOLOGY

1.Milestones

Enter the expected and actual dates for each milestone below, or N/A if the milestone does not apply to this project. If an expected date is an estimated date rather than a contract date, put an asterisk after that date.

| Milestone | Expected Date | Actual Date |
|---|---|---|
| Project Start Date | __/__/__ | __/__/__ |
| System Requirements Review (SRR) | __/__/__ | __/__/__ |
| Software Specification Review (SSR) | __/__/__ | __/__/__ |
| System Design Review (SDR) | __/__/__ | __/__/__ |
| System Hardware Preliminary Design Review | __/__/__ | __/__/__ |
| System Software Preliminary Design Review | __/__/__ | __/__/__ |
| System Software Critical Design Review | __/__/__ | __/__/__ |
| Test Readiness Review (TRR) | __/__/__ | __/__/__ |
| Functional Configuration Audit (FCA) | __/__/__ | __/__/__ |
| Physical Configuration Audit (PCA) | __/__/__ | __/__/__ |
| Formal Qualification Review (FQR) | __/__/__ | __/__/__ |
| Operational Test and Evaluation (OTE) | __/__/__ | __/__/__ |
| Project Completion Date | __/__/__ | __/__/__ |

2. Schedule

a. Select the schedule and staffing constraints which best describe this development:

[ ] Normal average schedule, effort, and quality
[ ] Shortest development schedule, extra staffing
[ ] Lowest cost with reduced staffing
[ ] Highest quality and reliability
[ ] Shortest schedule with high quality and reliability
[ ] Match staff size, with normal development
[ ] Match staff size, with shortest schedule
[ ] Match staff size, with very high quality

b. Schedule Aggressiveness (as % of nominal effort; i.e., if schedule is extremely curtailed, aggressiveness would be greater than 160%)

[ ] 75%
[ ] 100% (normal development schedule)
[ ] 130%
[ ] greater than 160%

## PROJECT QUESTIONNAIRE

### DEVELOPMENT METHODOLOGY

3. Development Standards

   a. Check all types of standard used in this development:

   [ ] Commercial [ ] IEEE [ ] Military
   [ ] Other [ ] None

   b. List the name(s) of these standard(s):

   ______________________________________

   c. Were these standards tailored specifically for use on this effort?

   [ ] Yes [ ] No

   d. Do you have a set of database standards and administrative procedures that were used in this development?

   [ ] Yes [ ] No

4. Development paradigm employed (check all that apply):

   [ ] Waterfall development [ ] Incremental development
   [ ] Phased builds [ ] Spiral development
   [ ] Rapid Prototyping [ ] Pilot development
   [ ] Other ______________________________________

5. Software Reviews

   a. Select all informal reviews held on the software during this development:

   [ ] Design walkthroughs
   [ ] Design inspections
   [ ] Code walkthroughs
   [ ] Code inspections
   [ ] Other ______________________________________

   b. Select all management reviews held on the software for this project:

   [ ] Monthly project reviews
   [ ] Weekly status reviews
   [ ] Other: ______________________________________

## PROJECT QUESTIONNAIRE

<u>DEVELOPMENT METHODOLOGY</u>, cont'd.

6. Documentation Requirements

   a. Number of documents required: _____

   b. Total number of pages of all documents: _____

   c. Select any of the following tools used:

      [ ] Design document generator
      [ ] Graphics generator
      [ ] Text automation
      [ ] Other ______________________________

   d. Documentation is primarily written by:

      [ ] Professional writers [ ] Programmers
      [ ] Other ______________________________

7. System/Software Requirements

   a. Select the option which corresponds to the level of definition and understanding of system requirements:

      [ ] Very little definition and understanding of system requirements
      [ ] Questionable definition and understanding of system requirements
      [ ] Fairly complete definition and understanding of system requirements
      [ ] Very complete definition and understanding of system requirements

   b. How will overall technology changes impact the project?

      [ ] During the development, the requirements will change more than once to upgrade the system, due to significant improvements in technology
      [ ] During the development, there will be at least on (but less than three) significant modifications to the system due to technology upgrades
      [ ] During the development there will be some minor modifications due to technology upgrades
      [ ] There will be no changes to the system or requirements during the development effort

DEVELOPMENT METHODOLOGY, cont'd.

c. Rate requirements volatility during development:

[ ] No changes
[ ] Small non-critical changes
[ ] Frequent non-critical changes
[ ] Occasional moderate changes
[ ] Frequent moderate changes
[ ] Many large changes

8. Design

a. Rate the maturity of the design concepts used:

| | |
|---|---|
| [ ] Experimental | [ ] Evolutionary |
| [ ] State-of-the-art | [ ] Mature |

b. Select any of the following design technologies, strategies, and tools used:

| | |
|---|---|
| [ ] Applications Generator | [ ] Object-oriented methods |
| [ ] CASE tools | [ ] Structured analysis |
| [ ] Relational database | [ ] Query Language (SQL) |
| [ ] 4GL or 5GL | [ ] Reuse libraries |
| [ ] Exception handling | [ ] Front loaded scheduling |
| [ ] Management toolset | [ ] Cost and schedule models |

[ ] Database management system
[ ] Small up-front design teams
[ ] Continuous integration via package specifications
[ ] Ada PDL
[ ] Other PDL ______________________________

c. Select the percentage of package specifications compiled successfully at PDR:

[ ] 20% [ ] 40% [ ] 60% [ ] 75% [ ] 90% [ ] 100%

d. Select the percentage of risks eliminated by PDR:

[ ] 20% [ ] 40% [ ] 60% [ ] 75% [ ] 90% [ ] 100%

## PROJECT QUESTIONNAIRE

DEVELOPMENT METHODOLOGY, cont'd.

9. Integration

   a. Rate the expected level of difficulty of integrating and testing the components to the CSCI level:

      [ ] No internal integration.
      [ ] Very little integration, no complex interfaces.
      [ ] Average degree of CSCI integration and interface complexity.
      [ ] Several CSCI interfaces with some complex interfaces.
      [ ] Complex, time-intensive CSCI integration process anticipated.

   b. Rate the expected level of difficulty of integrating and testing the CSCIs to the system level:

      [ ] Very little integration, no complex interfaces.
      [ ] Average degree of system integration and interface complexity.
      [ ] Several system interfaces with some complex interfaces.
      [ ] Complex, time-intensive system integration process anticipated.

10. Commercial off-the-shelf Software (COTS)

   a. Select the option which best describes the expected impact of integrating commercial off-the-shelf software into the system:

      [ ] There will be many impacts on the design/development effort to ensure that the vendor supplied COTS software will interface correctly with the developed operational software.
      [ ] There will be some impacts on the design/development effort to ensure that the vendor supplied COTS software will interface correctly with the developed operational software.
      [ ] There will be few impacts created by the COTS software packages to support the operating environment of the applications software.
      [ ] There will be no impacts caused by the purchased software as the purchased software only performs an operating environment support function (i,e., operating system).

**DEVELOPMENT METHODOLOGY**, cont'd.

11. Use of Software Tools

| | |
|---|---|
| [ ] Basic Ada language tools | [ ] MAPSE |
| [ ] Full, life cycle APSE | [ ] APSE |

## PROJECT QUESTIONNAIRE

### SOFTWARE SIZE

1. Size Estimates

   a. Number of major software subsystems/programs:_____

   b. Select the basis for size estimates:

   [ ] KSLOCs [ ] Carriage returns
   [ ] Function Points [ ] Semicolons
   [ ] Other:_______________________________

   c. Enter the requested sizing information below, in thousands of lines of source code (KSLOCs).

| Deliverable Program | New | Size Reused | Modified | Language |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

   d. Reused Design: ______%

   e. Amount of software packaged to be reused: ______%

## PROJECT QUESTIONNAIRE

### SOFTWARE SIZE

2. Source Code Mix

   a. CSCI Source Code Mix

| Source Code Type | % Code | % New Design | % New Code |
|---|---|---|---|
| Operating Systems............ | ______ | ______ | ______ |
| Interactive Operations....... | ______ | ______ | ______ |
| Real-Time Command & Control.. | ______ | ______ | ______ |
| On-Line Communications....... | ______ | ______ | ______ |
| Data Storage & Retrieval..... | ______ | ______ | ______ |
| String Manipulation.......... | ______ | ______ | ______ |
| Mathematical Operations...... | ______ | ______ | ______ |
| Other........................ | ______ | ______ | ______ |
| Other........................ | ______ | ______ | ______ |

   b. Statement Types

| | | | |
|---|---|---|---|
| Logical | ____% | Command | ____% |
| Data Typing | ____% | Declaration | ____% |
| Ada Tasking | ____% | Invocation | ____% |
| Mathematical | ____% | Data Manipulation | ____% |

3. Function Points

   If the system was sized using function points, please provide the following information:

   a. Number of inputs (unique major data types that enter the system):_____

   b. Number of outputs (unique logical major report formats the system will generate):_____

   c. Number of inquiries (types of queries that result in informational searches and responses):_____

   d. Number of external interfaces:_____

   e. Number of internal files (unique logical files/databases used by the application):_____

4. Database Size

   a. Database size, as a percentage of total program size: _____%

## PROJECT QUESTIONNAIRE

**PROJECT STAFFING**

1. Staff Size/Availability

   a. Total staff:______________(staff-months effort end-to-end)

   b. Minimum staff size:________

   c. Maximum staff size:________

   d. Staff Availability:________%
   (If all project personnel are full-time, enter 100%.)

2. Staff Skill/Experience

   a. Skill Level of Manager

      [ ] Bottom 15% [ ] 35% [ ] 55% [ ] 75% [ ] Top 90%

   b. Skill Level of Analysts

      [ ] Bottom 15% [ ] 35% [ ] 55% [ ] 75% [ ] Top 90%

   c. Skill Level of Programmers

      [ ] Bottom 15% [ ] 35% [ ] 55% [ ] 75% [ ] Top 90%

   d. Analysts' experience with similar applications: ______years
   ______months

   e. Select the option below that best describes the experience of the staff on this project:

      [ ] All experts in the type of program being developed
      [ ] Majority of experts but some new hires or novices
      [ ] Even mixture of experts and new hires or novices
      [ ] Majority of new hires or novices with few experts
      [ ] All personnel are new to this kind of program

   f. Programmers' experience with this host machine:
   _______years ______months

   g. Host Machine Expertise:

      [ ] Completely new hosting hardware system
      [ ] Mostly new hosting hardware system
      [ ] Most of the hardware system has been utilized by members of the development team before

## PROJECT QUESTIONNAIRE

PROJECT STAFFING, cont'd.

h. Software Language and Operating System Expertise:

- [ ] Completely new hosting hardware system
- [ ] Mostly new hosting hardware system
- [ ] Most of the hardware system has been utilized by the company before
- [ ] Extensive experience with hardware system

i. Analysts' experience with chosen development Methodology: _____years ______months

j. Analysts' experience with the Ada language: _______years ______months

k. Number of Ada Projects Completed by Team Members:_______

l. Analysts' experience with Ada Process Model:

- [ ] No familiarity with practices
- [ ] Little familiarity with practice
- [ ] Some familiarity with practices
- [ ] General familiarity with practices
- [ ] Successful on 1 mission critical project
- [ ] Successful on at least 1 mission critical project

m. Analysts' experience with Modern Programming Development Practices (MODP)

- [ ] No Use
- [ ] Beginning
- [ ] Reasonably Experienced in some MODP
- [ ] Reasonably Experienced in most MODP
- [ ] Routine Use of all MODP

n. Programmers' Ada Environment Experience

- [ ] Less than 3 months of experience
- [ ] Between 3 - 6 months of experience
- [ ] Between 6 - 12 months of experience
- [ ] Over 1 year of experience

<u>PROJECT STAFFING</u>, cont'd.

o. Company's experience developing this type of application
   - [ ] This application is a new project not in our current line of business
   - [ ] This application is a normal development project that is part of our current line of business
   - [ ] This application is a familiar type of project having already been developed by the company before or similar to other projects we have developed
   - [ ] Many applications of this type have been developed by the company (greater than 7)

3. General

a. Staffing will be done by

[ ] Developer
[ ] Developer and user

b. Enter the average work week for the staff on this project: _____ hours

c. Enter the normal work year within your company: _____ days

4. Teamwork Capability

a. Select the option which best represents the project analysts' performance as a team:

[ ] Non-Functional Team
[ ] Functional but not very effective
[ ] Functional and Effective
[ ] Extraordinary
[ ] Near Perfect Functioning

b. Select the option which best represents how well the programmers working on the project will perform as a team:

[ ] Non-Functional Team
[ ] Functional but not very effective
[ ] Functional and Effective
[ ] Extraordinary
[ ] Near Perfect Functioning

c. Select the type of team used for software development

| | |
|---|---|
| [ ] Design teams | [ ] Programming teams |
| [ ] Interdisciplinary teams | [ ] Participatory teams |

## PROJECT QUESTIONNAIRE

<u>COMPUTER SYSTEM</u>

1. Development Environment

   a. Configuration

      [ ] PC based
      [ ] Mainframe(s) with ASCII terminals
      [ ] Mainframe(s) with smart terminals
      [ ] Mainframe(s) with PC workstations
      [ ] Mainframe(s) with minus networked to workstations
      [ ] Mainframe(s) as part of networked homogeneous environment
      [ ] Mainframe(s) as part of networked heterogeneous environment
      [ ] Other:________

   b. Number of different types of workstation: _______(0 to 100)

   c. Rate the virtual machine volatility of the development system, based on frequency of major/minor changes:

      [ ] 12 months (major)/1 month (minor)
      [ ] 6 months/2 weeks
      [ ] 2 months/1 week
      [ ] 2 weeks/2 days

   d. Select the following option that best assesses the embedded features of the development system:

      [ ] Hardware is to be developed, but its completion will occur long before the software is to be ready
      [ ] Hardware is to be developed on the contract, it is to be developed concurrently with the software and the hardware can/does have major impacts on the software
      [ ] Hardware is to be developed on the contract, it is to be developed concurrently with the software but the hardware has little impact on the software
      [ ] No new hardware is to be developed under the effort; there will be no impact on the software development

## COMPUTER SYSTEM

e. Rate the software tool/environment stability of the development system:

[ ] Buggy compiler. APSE change every 2 weeks.
[ ] Stable but incapable compiler. APSE change every month. New tool rate 1 per week.
[ ] Stable compiler. APSE change every 3 months. New tool rate 1 per quarter.
[ ] Stable compiler. APSE change every 4 months. New tool rate 1 per month.
[ ] Stable compiler capable of tasking. APSE change every 6 months. New tool rate 1 per quarter.
[ ] Stable and fully capable compiler. APSE change every 6 months. New tool rate 1 per 6 months.

2. Target Computer Configuration (Complete this section only if the development system differs from the target.)

a. Rate the virtual machine volatility of the target system, based on number of major/minor changes:

[ ] 12 months (major)/1 month (minor)
[ ] 6 months/2 weeks
[ ] 2 months/1 week
[ ] 2 weeks/2 days

b. Identify the system architecture:

[ ] Centralized (single processor)
[ ] Tightly-coupled (multiple processor)
[ ] Loosely-coupled (multiple processor)
[ ] Functional processors communicating via a bus
[ ] Distributed (centralized database)
[ ] Distributed (distributed database)

c. Rehosting Impact (effort to convert from host to target system)

[ ] None
[ ] Minor language and/or system change
[ ] Major language or system change
[ ] Major language and system change

## PROJECT QUESTIONNAIRE

COMPUTER SYSTEM, cont'd.

3. Performance Requirements

   a. Main Storage Constraint. Main storage refers to direct random access storage such as core, integrated-circuit, or plated-wire storage; it excludes such devices as drums, disks, tapes, or bubble storage. Select the percentage which best reflects the percentage of main storage expected to be used by the subsystem and any other subsystems consuming the main storage resources.

      [ ] at most 50%
      [ ] 70%    [ ] 85%    [ ] 95%

   b. Overall Hardware Constraints. Overall hardware refers to processor memory, speed, and throughput

      [ ] Close to 100% utilization
      [ ] Difficult hardware capacity limitations (75% to 90%)
      [ ] Average hardware capacity limitations (60% to 75%)
      [ ] No hardware capacity limitations (60% to 75%)

   c. Execution Time Constraints. Select the percentage which best reflects the percentage of available execution time expected to be used by the subsystem and any other subsystems consuming the execution time resource.

      [ ] at most 50%
      [ ] 70%    [ ] 85%    [ ] 95%

   d. Interactive Response Time

      [ ] subsecond
      [ ] 1-5 seconds
      [ ] 5-10 seconds
      [ ] greater than 10 seconds
      [ ] Response time is not a factor

4. Microprocessor Code

   a. Percentage of software functions that are to be implemented in firmware: _____%

## PROJECT QUESTIONNAIRE

### DEVELOPMENT ENVIRONMENT

1. Project Organization

   a. Check all of the company organizations supported by the software organization

| | |
|---|---|
| [ ] Systems engineering | [ ] User department |
| [ ] Marketing | [ ] Software development |
| [ ] Program management | [ ] Software test |
| [ ] Configuration management | [ ] Quality assurance |
| [ ] Data management | [ ] Independent V&V |
| [ ] Database administration | |
| [ ] Other ____________________ | |

   b. Number of organizations within the company significantly involved during the software development: ______

   c. Organizational Interface Complexity

      [ ] Single customer, single interface
      [ ] Single customer collocated with developer
      [ ] Multiple internal interface single external interface
      [ ] Multiple internal and external interfaces
      [ ] Multiple interfaces geographically distributed

   d. Multiple site development

      [ ] Single site and single organization
      [ ] Single site and multiple organizations
      [ ] Multiple sites, same general location
      [ ] Multiple sites, located 50 miles or more apart

   e. Total size (in 100s) of all involved organizations, not just project personnel, involved in the project:_______

   f. Number of locations at which software is developed (from 1 to 100):______

   g. Number of company locations that must be travelled to for project work:______

   h. Number of customer locations: ______

PROJECT QUESTIONNAIRE

DEVELOPMENT ENVIRONMENT

i. Select the commuting rate that characterizes the overall commuting rate of project personnel:

[ ] Normal (less than 1 hour each way)
[ ] Significant (more than 1 hour each way)
[ ] Physical relocation of project team for part of the work

j. Number of physical personnel moves required between locations or departments:_______

2. Computer Resources

a. Select the percentage of time that the development computer is available for use on this project:

[ ] 10% [ ] 40% [ ] 70% [ ] 100% (fully dedicated)

b. Computer terminal allocation to development team:

[ ] 1 terminal/person
[ ] 1 terminal/2 persons
[ ] 1 terminal/> 2 persons

c. Select the average time required to log on, edit, compile, link, execute, and receive a hardcopy

[ ] < 4 hours [ ] 4-12 hours [ ] > 12 hours

3. Office Facilities

a. Software Development Office Facilities

[ ] Individual Offices
[ ] 2-person offices
[ ] Multi-employee offices
[ ] Open/no private offices
[ ] Cramped

b. Security Requirements

[ ] Unclassified
[ ] Unclassified/Secure
[ ] Classified
[ ] Classified/physically Secure

DEVELOPMENT ENVIRONMENT, cont'd.

4. System User Profile

a. Does the user already perform the proposed function?

[ ] No [ ] Yes, manually [ ] Yes, automated

b. User's experience with related applications:

[ ] Low [ ] Average [ ] High

c. How will the user's data processing knowledge impact the project?

[ ] Help [ ] Hinder

# APPENDIX H

## PROJECT DESCRIPTIONS

This appendix contains project descriptions of the eight projects targeted in the test case study. The information is presented according to the following outline:

- I. Project Description
    - Application Type
    - Complexity
    - Testing Required
    - Displays
    - Required Reliability
- II. Software Size
    - ASLOC
        - New
        - Reused
        - Counting Convention
    - Other
        - New
        - Reused
    - % for Reuse
- III. Development Process
    - Methodologies
    - Requirements
    - Tools Available
    - Standards Used
- IV. Computer System
    - Type
    - Workstation Types
    - Hardware Developed in Parallel
    - Target vs. Host
    - Interactive Response Time

PROJECT 1

PROJECT DESCRIPTION

APPLICATION TYPE: Command and Control

COMPLEXITY: Some Difficult or Complex Calculations
Tasking and Generics Used

TESTING REQUIRED: Normal Testing by Developer

DISPLAYS: User-Friendly
Menu Driven Displays

REQUIRED RELIABILITY: Special Backup Considerations to Ensure no Failure

SOFTWARE SIZE

SLOC: 50,000 New Ada

COUNTING CONVENTION: Terminal Semicolons

% FOR REUSE: 10% Packaged for Reuse in any Other Application

DEVELOPMENT PROCESS

METHODOLOGIES: Prototyping
Incremental Development
Object Oriented Design Plus Structured Analysis
Continuous Integration via Package Specs
Monthly Project Reviews

REQUIREMENTS: Fairly Complete System Requirements
Occasional Moderate Requirement Changes

TOOLS AVAILABLE: Basic Ada Language Tools
Documentation Tools

STANDARDS USED: Commercial Standards

## COMPUTER SYSTEM

TYPE: Distributed (Distributed Database)

WORKSTATION TYPES: 2

HARDWARE DEVELOPED IN PARALLEL: No

TARGET VS. HOST: Same Machine, No Change Required

INTERACTIVE RESPONSE TIME: 1-5 Seconds

## PROJECT STAFFING

RATING:

MANAGERS - Above Average (75%)
ANALYSTS - Above Average (75%)
PROGRAMMERS - Above Average (75%)

TEAM COMPOSITION: Even Mixture of Experts and New Hires. All Have Had Some Background in the Ada Language.

YEARS EXPERIENCE WITH MACHINE: 2 Years

YEARS EXPERIENCE WITH ADA: 1 Year

YEARS EXPERIENCE WITH METHODOLOGY: 1 Year

EXPERIENCE WITH MODP: Reasonably Experienced with Some

TYPE OF TEAMS USED: Participatory Teams

TYPE OF PROJECT FOR STAFF: Normal New Project

STAFF AVAILABILITY: [illegible]

## DEVELOPMENT ENVIRONMENT

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 1

PERCENT OF COMPUTER RESOURCES AVAILABLE: 100% (Fully Dedicated)

RATIO OF TERMINALS TO PERSONNEL: 1 Per Person

SECURITY REQUIREMENTS: Unclassified

## PROJECT 2

### PROJECT DESCRIPTION

| | |
|---|---|
| APPLICATION TYPE: | Command and Control |
| COMPLEXITY: | Difficult Highly Nested Logic, Real-Time Processing |
| TESTING REQUIRED: | Normal |
| DISPLAYS: | Interactive with Pointing Devices |
| REQUIRED RELIABILITY: | None |

### SOFTWARE SIZE

| | |
|---|---|
| ASLOC: | 35,000 New Ada |
| OTHER: | 42,000 New Assembly<br>38,000 Reused Assembly |
| COUNTING CONVENTION: | Terminal Semicolons |
| % FOR REUSE: | 10% Package for Reuse in any other application |

### DEVELOPMENT PROCESS

| | |
|---|---|
| METHODOLOGIES: | Evolutionary Development<br>Object Oriented Design with Structured Analysis<br>Design and Code Walkthroughs<br>Phased Builds |
| REQUIREMENTS: | Fairly Complete Understanding of Requirements<br>Many Major Changes |
| TOOLS AVAILABLE: | Basic Ada Language Tools<br>Document and Design Tools |
| STANDARDS USED: | MIL-STD 1679 483 188<br>MIL-STD 482 490 |

COMPUTER SYSTEM

TYPE: Distributed

WORKSTATION TYPES: 1

HARDWARE DEVELOPED IN PARALLEL: Yes

TARGET VS. HOST: Major Language or System Change

INTERACTIVE RESPONSE TIME: 1-5 Seconds

PROJECT STAFFING

RATING:

| | |
|---|---|
| MANAGERS - | Not Available |
| ANALYSTS - | Average |
| PROGRAMMERS - | Average |

TEAM COMPOSITION: Even Mixture of Experts and New Hires

YEARS EXPERIENCE WITH MACHINE: 12 Years

YEARS EXPERIENCE WITH ADA: 2 Years

YEARS EXPERIENCE WITH METHODOLOGY: 0.5 Years

EXPERIENCE WITH MODP: Reasonably Experienced with Some

TYPE OF TEAMS USED: Functional Teams with Leader

TYPE OF PROJECT FOR STAFF: Normal New Development

STAFF AVAILABILITY: 100%

DEVELOPMENT ENVIRONMENTS

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 3

PERCENT OF COMPUTER RESOURCES AVAILABLE: 100%

RATIO OF TERMINALS TO PERSONNEL: 1

SECURITY REQUIREMENTS: Unclassified

Additional 20 Months Due to Having to Change Ada Compilers, Late Compiler Deliveries, Late Target Computer Memory, Difficult Integration

**PROJECT 3**

PROJECT DESCRIPTION

APPLICATION TYPE: Environment/Tools
Production Center, Contracted Software

COMPLEXITY: Routine Nesting, Minimal Interface with Operating System, Standard I/O

TESTING REQUIRED: Normal Testing by Developer

DISPLAYS: Simple Inputs/Outputs

REQUIRED RELIABILITY: None

SOFTWARE SIZE

ASLOC: 13,600 New Ada
5040 Reused Ada

COUNTING CONVENTION: Terminal Semicolons

% FOR REUSE: 10% to be packaged for reuse

DEVELOPMENT PROCESS

METHODOLOGIES: Object Oriented
Ada PDL
Bottom Up Development
Structured Analysis
Chief Programmer for Team Development Strategies
Code Walkthroughs

REQUIREMENTS: Fairly Complete Definition and Understanding
System Requirements
Frequent Noncritical Changes

TOOLS AVAILABLE: Requirements Analysis
Design Documentation Generation

STANDARDS USED: MIL-STD 1679

## COMPUTER SYSTEM

TYPE: Distributed

WORKSTATION TYPES: 1

HARDWARE DEVELOPED IN PARALLEL: Yes

TARGET VS. HOST: None Required

INTERACTIVE RESPONSE TIME: 1-5 Seconds

## PROJECT STAFFING

RATING:

MANAGERS - Not Available
ANALYSTS - Average
PROGRAMMERS - Above Average

TEAM COMPOSITION: Team Ranged from a Master's Experience in Computer Science plus 4 Years Job Experience

YEARS EXPERIENCE WITH MACHINE: 1 Year

YEARS EXPERIENCE WITH ADA: 2 Years

YEARS EXPERIENCE WITH METHODOLOGY: 0 Years

EXPERIENCE WITH MODP:

TYPE OF TEAMS USED: Programming Teams/Chief Programmer

TYPE OF PROJECT FOR STAFF: Normal New Product

STAFF AVAILABILITY: 70%

## DEVELOPMENT ENVIRONMENTS

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 1

PERCENT OF COMPUTER RESOURCES AVAILABLE: 100%

RATIO OF TERMINALS TO PERSONNEL: 1 Per Person

SECURITY REQUIREMENTS: Unclassified

EXTRA: Three Month Hiatus

**PROJECT 4**

PROJECT DESCRIPTION

APPLICATION TYPE: Environmental/Tools Prototype
Production Center, Internally Developed

COMPLEXITY: Routine Nesting, Minimal Interface with Operating System, Standard I/O

TESTING REQUIRED: Normal Testing by Developer

DISPLAYS: Simple Inputs/Outputs

REQUIRED RELIABILITY: None

SOFTWARE SIZE

ASLOC: 480,000 New Ada

COUNTING CONVENTION: Terminal Semicolons

% FOR REUSE: 10% Packaged for Reuse

DEVELOPMENT PROCESS

METHODOLOGIES: Object Oriented Design
Rapid Prototyping
Independent Teams working in Parallel
Ada PDL
Incremental Development
Compilable Package Specs by PDR
** Very Familiar with Ada Design Practices

REQUIREMENTS: Fairly Complete Requirements
Noncritical Requirement Changes

TOOLS AVAILABLE: Document Generator
APSE

STANDARDS USED: MIL-STD 1815
Not Much Documentation Required Since Prototype

## COMPUTER SYSTEM

TYPE: Distributed

WORKSTATION TYPES: 1

HARDWARE DEVELOPED IN PARALLEL: No

TARGET VS. HOST: No Change

INTERACTIVE RESPONSE TIME: 1-5 Seconds

## PROJECT STAFFING

RATING:

MANAGERS - 98% Top Notch
ANALYSTS - 98% Top Notch
PROGRAMMERS - 98% Top Notch

TEAM COMPOSITION: ** All Experts
Team Composed of Ada Experts

YEARS EXPERIENCE WITH MACHINE: 5 Years

YEARS EXPERIENCE WITH ADA: 5 Years

YEARS EXPERIENCE WITH METHODOLOGY: 5 Years

EXPERIENCE WITH MODP: Reasonably Experienced

TYPE OF TEAMS USED: Interdisciplinary Teams

TYPE OF PROJECT FOR STAFF: Normal Project

STAFF AVAILABILITY: 100%

## DEVELOPMENT ENVIRONMENTS

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 1

PERCENT OF COMPUTER RESOURCES AVAILABLE: 100%

RATIO OF TERMINALS TO PERSONNEL: 1

SECURITY REQUIREMENTS: Unclassified

PROJECT 5

PROJECT DESCRIPTION

APPLICATION TYPE: Environment/Tools

COMPLEXITY: Difficult Highly Nested Logic, Real-Time Processing

TESTING REQUIRED: Normal Testing by Developer

DISPLAYS: User-Friendly, Menu Driven
Graphics Oriented

REQUIRED RELIABILITY: None

OTHER:

SOFTWARE SIZE

ASLOC: 136,000 New Ada

COUNTING CONVENTION: Physical lines (includes blank lines and comments)

OTHER: None

% FOR REUSE: 10%

DEVELOPMENT PROCESS

METHODOLOGIES: Incremental Development
Phased Builds
Rapid Prototyping
Pilot Development
Small Up-front Design Teams
Object-Oriented Methods
Structured Analysis
Design and Code Walkthroughs

REQUIREMENTS: Questionable Definition and Understanding of System Requirements
Occasional Moderate Changes to Software Requirements

TOOLS AVAILABLE: Basic Ada Language Tools
Design Documentation Generator
Graphics Generator

STANDARDS USED: MIL STD 2167A

## COMPUTER SYSTEM

TYPE: Functional Processors Communicating via a Bus

WORKSTATION TYPES: 3

HARDWARE DEVELOPED IN PARALLEL: No

TARGET VS. HOST: Major Language or System Change Required for Rehosting

INTERACTIVE RESPONSE TIME: 1-5 Seconds

## PROJECT STAFFING

RATING:

MANAGERS - Top 90%
ANALYSTS - Above Average
PROGRAMMERS - Above Average

TEAM COMPOSITION: Experience Ada Personnel

YEARS EXPERIENCE WITH MACHINE: 5 Years

YEARS EXPERIENCE WITH ADA: 3 Years

YEARS EXPERIENCE WITH METHODOLOGY: 3 Years

EXPERIENCE WITH MODP: Reasonably Experienced in Most

TYPE OF [illegible]AMS USED: Design, Programming, Participatory Interdisciplinary Teams

TYPE OF PROJECT FOR STAFF: Normal Project

STAFF AVAILABILITY: 75%

## DEVELOPMENT ENVIRONMENTS

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 1

PERCENT OF COMPUTER RESOURCES AVAILABLE: 100%

RATIO OF TERMINALS TO PERSONNEL: 1 Terminal/Person

SECURITY REQUIREMENTS: Unclassified/Secure

MICROCODE: 50%

PROJECT 6

PROJECT DESCRIPTION

APPLICATION TYPE: Avionics (Guidance and Control)

COMPLEXITY: Difficult Highly Nested Logic,
Real-time Processing

TESTING REQUIRED: Normal Testing by Developer

DISPLAYS: None

REQUIRED RELIABILITY: None

SOFTWARE SIZE

ASLOC: 19,200 New Ada, 7,200 Reused

OTHER: 4800 New Assembly
600 Reused Assembly

COUNTING CONVENTION: Terminal Semicolons

% FOR REUSE: 10%

DEVELOPMENT PROCESS

METHODOLOGIES: Incremental Development
Phased Builds
Small-Up-Front Design Team
More Time Spent in Design, Less Time in Code
Integration and Testing
Structured Analysis
Continuous Integration via Package Specs
Compilable Package Specs by PDR

REQUIREMENTS: Fairly Complete Definition and Understanding of
System Requirements
Many Major Changes Occurred with Software
Requirements

TOOLS AVAILABLE: Basic Ada Language Tools
No Documentation or Design Tools Available

STANDARDS USED: MIL-STD 1679, 483, 490

COMPUTER SYSTEM

TYPE: Centralized (Single Processor)

WORKSTATION TYPES: 1

HARDWARE DEVELOPED IN PARALLEL: None

TARGET VS. HOST: Major Language or System Change Required for Rehosting

INTERACTIVE RESPONSE TIME: Not Applicable

PROJECT STAFFING

RATING:

MANAGERS - Top Percent (Excellent)
ANALYSTS - Above Average
PROGRAMMERS - Above Average

TEAM COMPOSITION: Experts and Some New Hires

YEARS EXPERIENCE WITH MACHINE: 0 Years

YEARS EXPERIENCE WITH ADA: 0 Years

YEARS EXPERIENCE WITH METHODOLOGY: 7 Years

EXPERIENCE WITH MODP: Reasonably Experienced in Some MODP

TYPE OF TEAMS USED: Participatory Teams

TYPE OF PROJECT FOR STAFF: Normal New Project

STAFF AVAILABILITY: 100%

DEVELOPMENT ENVIRONMENTS

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 1

PERCENT OF COMPUTER RESOURCES AVAILABLE: 100%

RATIO OF TERMINALS TO PERSONNEL: 1 Per Person

SECURITY REQUIREMENTS: Unclassified

PROJECT 7

PROJECT DESCRIPTION

APPLICATION TYPE: Command and Control

COMPLEXITY: Difficult Highly Nested Logic
Real-Time Processing

TESTING REQUIRED: Normal Testing by Developer

DISPLAYS: Mostly Simple Displays but a Small Part is Graphics Oriented

REQUIRED RELIABILITY: None

SOFTWARE SIZE

ASLOC: 69,160

OTHER: None

COUNTING CONVENTION: Terminal Semicolons

% FOR REUSE: 10%, of software to be packaged for reuse

DEVELOPMENT PROCESS

METHODOLOGIES: Incremental Development
Small Up Front Design Teams
More Time Spent in Design, Less Time in Code, Integration and Testing
Structured Programming
Continuous Integration via Package Specs
Code Walkthroughs and Inspections
Design Walkthroughs and Inspections

REQUIREMENTS: Questionable Definition of System Requirements
Occasional or Moderate Software Requirement Changes

TOOLS AVAILABLE: Basic Ada Language Tools
Documentation tools

STANDARDS USED: Tailored MIL-STD 480, 483, 490, 1521
MIL-S-52779

## COMPUTER SYSTEM

TYPE: Distributed System (Distributed Database)

WORKSTATION TYPES: 0 Workstations

HARDWARE DEVELOPED IN PARALLEL: Yes

TARGET VS. HOST: Same Machine, No Change Required

INTERACTIVE RESPONSE TIME: 1-5 Seconds

## PROJECT STAFFING

RATING:

MANAGERS - Below Average (35%)
ANALYSTS - Average (55%)
PROGRAMMERS - Average (55%)

TEAM COMPOSITION: Even Mixture of Experts and New Hires

YEARS EXPERIENCE WITH MACHINE: 0 Years

YEARS EXPERIENCE WITH ADA: 0 Years

YEARS EXPERIENCE WITH METHODOLOGY: 0 Years

EXPERIENCE WITH MODP: Reasonably Experienced in Some

TYPE OF TEAMS USED: Design and Programming Teams

TYPE OF PROJECT FOR STAFF: Although Company had a lot of Experience in this field, for this Division it was First of its Kind

STAFF AVAILABILITY: 100%

## DEVELOPMENT ENVIRONMENTS

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 1

PERCENT OF COMPUTER RESOURCES AVAILABLE: 80%

RATIO OF TERMINALS TO PERSONNEL: 1 Per Person

SECURITY REQUIREMENTS: Unclassified

PROJECT 8

PROJECT DESCRIPTION

| | |
|---|---|
| APPLICATION TYPE: | Command and Control<br>Component within a System |
| COMPLEXITY: | Complex Dynamic Resource Allocation<br>Multiple Exception Handlers |
| TESTING REQUIRED: | Normal Testing By Developer |
| DISPLAYS: | Pressure Sensitive Devices |
| REQUIRED RELIABILITY: | None |

SOFTWARE SIZE

| | |
|---|---|
| ASLOC: | 16,470 |
| OTHER: | 1,830 Assembly |
| COUNTING CONVENTION: | Terminal Semicolons |
| % FOR REUSE: | 15% of Software Packaged for Reuse |

DEVELOPMENT PROCESS

| | |
|---|---|
| METHODOLOGIES: | Waterfall Development<br>Phased Builds<br>Object Oriented Methods Plus Structured Programming<br>Monthly Project Reviews<br>Exception Handlers<br>Compilable Package Specs by PDR |
| REQUIREMENTS: | Fairly Complete System Requirements<br>Many Major Software Requirements Changes |
| TOOLS AVAILABLE: | Basic Ada Language Tools<br>No Documentation Tools |
| STANDARDS USED: | Tailored MIL-STD 1679A<br>MIL-S-52779A |

COMPUTER SYSTEM

TYPE: Mainframe with ASCII Terminal
Functional Processors Communicating via Bus

WORKSTATION TYPES: 0

HARDWARE DEVELOPED IN PARALLEL: Yes

TARGET VS. HOST: Different Machines
Major Language and System Change Required

INTERACTIVE RESPONSE TIME: Subsecond

PROJECT STAFFING

RATING:

MANAGERS - Above Average (75%)
ANALYSTS - Above Average (75%)
PROGRAMMERS - Average (55%)

TEAM COMPOSITION: Even Mixture of Experts and New Hires

YEARS EXPERIENCE WITH APPLICATION: 5 Years

YEARS EXPERIENCE WITH MACHINE: 3 Years

YEARS EXPERIENCE WITH ADA: 0 Years

YEARS EXPERIENCE WITH METHODOLOGY: 2 Years

EXPERIENCE WITH MODP: Beginning

TYPE OF TEAMS USED: Design and Programming Teams

TYPE OF PROJECT FOR STAFF: Normal New Project

STAFF AVAILABILITY: 100%

DEVELOPMENT ENVIRONMENTS

NUMBER OF SITES DEVELOPMENT OCCURRED ON: 1

PERCENT OF COMPUTER RESOURCES AVAILABLE: 70%

RATIO OF TERMINALS TO PERSONNEL: 3 Per Person

SECURITY REQUIREMENTS: Unclassified

This page is intentionally left blank.

## REFERENCES

| | |
|---|---|
| [BOEH87] | Boehm, Barry, Walker Royce, TRW, "Ada COCOMO: TRW IOC Version", Proceedings Third COCOMO Users Group Meeting, Software Engineering Institute, Nov. 1987. |
| [BOEH81] | Boehm, Barry, Software Engineering Economics, Prentice Hall, 1981. |
| [BOEH88] | Boehm, Barry, Software Cost Estimation Using COCOMO, TRW, June 1988. |
| [BOEH88A] | Boehm, Barry, TRW, Dec 1988. |
| [BOOC87] | Booch, Grady, Software Engineering with Ada, The Benjamin/Cummings Publishing Company, Inc., 1987. |
| [CEI86] | Computer Economics, Inc., CEI Presents System-3, 1986. |
| [CHEA89] | Cheadle, William G., Martin Marietta Astonautics Group, Feb. 1989. |
| [GALO86] | Galorath, Daniel D., CEI, "Short and Long Term Ada Impacts Estimated with CEI System-3", Apr. 1986. |
| [HAD88] | Hadlock, Wayne W., National Sales Manager, Software Productivity Research, Inc., Dec 1988. |
| [IITR87] | IIT Research Institute, U.S. Army Cost and Economic Analysis Center Software Cost Model Research Paper, Sept. 1987. |
| [MART88] | Martin Marietta Denver Aerospace Corporation, SASET User's Guide, July 1988. |
| [ORME86] | Orme, Tony, "Project Management Experience of Ada", 1986. |
| [PARK89] | Park, Robert E., PRICE Systems, "Ada Estimating -A PRICE S Profile", Jan. 1989. |
| [PRIC87] | RCA Corporation, PRICE S User's Manual, 1987. |
| [RCI88] | Reifer Consultants, Inc., Softcost-Ada Users Manual Version 1.3, July 1988. |
| [RCI89] | SoftCost-Ada Users Group Meeting, Feb. 1989. |
| [REIF87] | Reifer, Donald J., Reifer Consultants, Inc., |

"Ada's Impact: A Quantitative Assessment", Sept. 1987.

[REIF89] Reifer, Donald J., Reifer Consultants, Inc., "Final Report, Ada Data Analysis and Normalization", Jan. 1989.

[SPR86] Software Productivity Research, Inc., User's Guide SPQR/20, 1986.

[STAN88] Stanley, Wayne, Sales Manager, Computer Economics, Inc., Dec. 1988.

[TECO87] Brenner, Neal J., Daniel D. Galorath, David G. Lawrence, Judy C. Rampton, Tecolote, "A Plan for Collecting Ada Software Development Cost, Schedule, and Environment Data", Apr. 1987.